National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 86/04)

Canada

The University of Alberta

REPRESENTATION AND SELECTION TECHNIQUES FOR GENETIC LEARNING SYSTEMS

by

Dale Eric Schuurmans

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Fall, 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-45632-9

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Dale Eric Schuurmans

TITLE OF THESIS: Representation and Selection Techniques for Genetic Learning Systems

DEGREE FOR WHICH THIS THESIS WAS PRESENTED: Master of Science

YEAR THIS DEGREE GRANTED: 1988

Permission is hereby granted to The University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) Dale Schuurmans

Permanent Address:
215, 10511 - 19 Avenue
Edmonton, Alberta
Canada T6J 5S8

Dated August 25/88

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **Representation and Selection Techniques for Genetic Learning Systems** submitted by **Dale Eric Schuurmans** in partial fulfillment of the requirements for the degree of **Master of Science.**

.....
Supervisor

.....

.....

.....

Date August 25/88

Dedicated to

Sharon Schuurmans

and to

Walter and Mary Schuurmans

for their patience, love and encouragement

ABSTRACT

Genetic learning systems are a form of automated discovery system that emulate natural processes of evolution. These systems have recently seen growing research interest and a wide variety of applications. A genetic learning system is comprised of a classifier system-based performance system and genetic algorithm-based learning mechanism.

Classifier systems are a form of rule-based production system incorporating a simplified representation scheme. However, the apparent simplicity of these systems is misleading. It is shown that certain limitations in the expressiveness of the classifier representation can create the potential for a combinatorial explosion in the number of classifiers needed to represent solutions to problems. This is especially problematic because the genetic algorithm does not discover multiple classifier solutions in an effective manner. The factors that contribute to this inability are examined.

Genetic learning systems must always balance the competing needs of creating new classifiers and exploiting the current classifier population. The "2-Armed Bandit Problem" expresses this fundamental exploration/exploitation tradeoff in its basic form. From a careful examination of this problem, a new selection technique is developed from statistical principles. This "likelihood-based" approach demonstrates superior convergence properties over the traditional "economics-based" approach. Experimental results show that many of the traditional weaknesses of economics-based selection procedures are overcome by this new procedure.

The result of this research is a domain-independent learning system that demonstrates a number of advantages over traditional genetic learning systems.

Acknowledgements

I would like to thank Jonathan Schaeffer, my supervisor, for his guidance, support, and optimism. Jonathan's help and encouragement contributed a great deal to the timely completion of this thesis research.

I would also like to thank the members of the examining committee: Renee Elio, Hong Zhang, and Curtis Strobeck for their helpful comments and suggestions. This thesis has benefited considerably from their valuable input.

Special thanks to Ursula Maydell for her help during the completion of the final draft of this thesis.

Table of Contents

Chapter	Page
Chapter 1: Introduction	1
Chapter 2: An Overview of Classifier Systems and Genetic Algorithms	10
2.1. Classifier Systems	10
2.2. Genetic Algorithms	16
2.2.1. Genetic Operators	17
2.2.2. Selection Processes	19
2.2.3. Feedback and Fitness	21
2.2.4. The Bucket Brigade Algorithm	23
2.2.5. Performance	24
2.2.6. External Parameters	25
Chapter 3: Difficulties with Classifier Representations	27
3.1. Human Factors	28
3.2. Attribute-Value Representation	28
3.3. Illegal Patterns	28
3.4. Classifier Expressibility	31
3.4.1. Disjunction	31
3.4.2. Position Dependent Semantics	34
3.4.3. Multiple Conditions and Parameterization	37
3.5. Remarks	38
Chapter 4: Discovering Classifier Solutions with Genetic Algorithms	40
4.1. Convergence	41
4.2. Interference	43
4.3. Independence	45
4.4. Discovering Chains	46
4.4.1. Sparse Feedback	47
4.4.2. Classifier Interdependence	49
4.5. Remarks	51
Chapter 5: A Reconsideration of Fitness and Selection	52
5.1. Feedback	53
5.2. Economic Fitness	56
5.3. Selection	60
5.3.1. The 2-Armed Bandit Problem	61
5.3.2. Asymptotically Optimal Stochastic Procedures	64
5.3.3. Convergence and Finite Optimality	67
5.3.4. A Fast Algorithm for the 2ABP	67

5.3.5. Generalization: The K-Armed Bandit Problem 76
5.4. Evaluating Fitness 80
5.5. Remarks 86
Chapter 6: Implementation and Experimental Results 87
6.1. Credit Assignment 87
6.1.1. Bernoulli Trial Sets 88
6.1.2. Parallel Testing 89
6.1.3. Limitations 90
6.1.4. Multidimensional Feedback 90
6.2. Selection Algorithms 91
6.2.1. Conflict Resolution 91
6.2.2. Reproduction 92
6.2.3. Extinction 93
6.3. `Tic-Tac-Toe` Legal Move Experiments 94
6.4. Extensions 98
6.4.1. Ubiquity 98
6.4.2. Dominance 101
6.4.3. Sub-Population Separation 104
Chapter 7: Concluding Remarks 107
References 112
Appendix 1: Procedures for the 2ABP and KABP Experiments 114
Appendix 2: Procedures for the `Tic-Tac-Toe` Legal Move Experiments 116

List of Tables

Table	Page
6.1 Legal Move Experiment with Economic Selection	96
6.2 Legal Move Experiment with Likelihood Selection	9
6.3 Legal Move Experiment with Ubiquity Bias	100
6.4 Legal Move Experiment with Dominance Bias	103
6.5 Legal Move Experiment with Sub-Populations	105

List of Figures

Figure	Page
5.1 2ABP Experiment with Economic Selection	74
5.2 2ABP Experiment with Likelihood Selection	74
5.3 KABP Experiment with Economic Selection	78
5.4 KABP Experiment with Likelihood Selection	78
5.5 2ABP Scaling Experiments with Econofnic Selection	82
5.6 2ABP Scaling Experiments with Likelihood Selection	82
5.7 2ABP Time Dependence Experiment with Economic Selection	84
5.8 2ABP Time Dependence Experiment with Likelihood Selection	84
6.1 Legal Move Experiment - Economic vs. Likelihood	95
6.2 Legal Move Experiment - Likelihood vs. Ubiquity	99
6.3 Legal Move Experiment - Ubiquity vs. Dominance	102
6.4 Legal Move Experiment - Dominance vs. Sub-Populations	105

Chapter 1

Introduction

Genetic algorithms have recently become an expanding area of research into computational models of learning and adaptation. As the name suggests, these learning models are inspired by natural processes of evolution. The idea that domain-independent machine learning systems could be synthesized from some of the basic principles of population genetics is one that has recently gained increasing attention. This approach saw development into its first practical forms with the early investigations of John Holland who laid many of the basic mathematical foundations and justifications underlying the basic learning technique [Hol75].

Although these learning algorithms were intended for application in a wide range of machine learning problems, until recently, their use was restricted predominantly to function/combinatorial optimization problems. However, with the recent advent of *classifier systems*, more substantial learning problems have been tackled [HHN86, Hol86]. Classifier systems are a simple form of rule-based production system that are well suited for manipulation by genetic algorithms. As such, classifier systems provide genetic learning systems with a computational component to serve as a task performance system. The genetic algorithm manipulates the structures of this performance system in an attempt to improve its capabilities in the task domain. It is this form of domain-independent machine learning system, comprised of a classifier based performance system and a genetic algorithm based adaptive component, that is the object of investigation of this thesis.

An intriguing aspect of these genetic learning systems is that they often begin with a *tabula rasa*. That is, no domain specific knowledge is possessed by the system before the learning task begins. In this respect, genetic algorithms can be thought of as *black box* learning algorithms.

Black Box Learning Systems

A black box learning system can be best thought of as an autonomous adaptive agent situated in an

environment. The environment could be in a number of states; some of which tend to be beneficial to the agent, and others which tend to be to the agent's detriment. Here the learning task belongs solely to the black box, it must autonomously determine courses of action which will place the environment increasingly into states which are to its benefit while avoiding detrimental states. A key feature of black box adaptive agents is that they may begin the learning task with a minimum of domain-dependent information. This minimum amount of information is characterized by the minimal requirements of the agent-environment coupling that still allow for the agent to demonstrate adaptive behavior in the environment.

- The agent must possess perceptual capabilities in the environment.
- The agent must possess effective behavioral capabilities in the environment.
- The environment must periodically provide the agent with feedback that somehow indicates the benefit or detriment that the current state holds for the agent.

An agent that is not able to sense relevant aspects of the environmental states it encounters will not be able to distinguish those states where a course of action might lead to a beneficial situation from those states where that same course of action will lead to a detrimental situation. Without adequate perceptual capabilities, an agent is afforded no basis upon which decisions about the desirability of different courses of action to take in particular situations could be made. Such an agent, if unable to distinguish those states when a particular course of action is desirable from those states where it is not, has no opportunity to adapt its decisions in a manner that would improve its performance in the environment.

An agent that cannot affect the environment, even minimally, to the extent that it can tend to avoid detrimental states and attain beneficial states, is afforded no opportunity to improve its performance, even if it can distinguish desirable from undesirable states. If none of the agent's available actions can influence the future proportion of beneficial and detrimental states it will encounter, then it is pointless for the agent to attempt to adapt its behavior.

Feedback is the only mechanism through which a black box learning system can differentiate the

benefits of the various states of the environment, consequently allowing the system to differentiate the benefits of various actions in particular states based upon the desirability of the states that result. Without feedback, the adaptive agent is left with no way to distinguish the desirability of one action from any another. In which case, it would not matter to the agent which of the alternate possible actions is taken in any particular situation. With no way to evaluate its choices, a learning system has no opportunity to modify its performance in anything other than a random, meaningless way. Ultimately, feedback defines the goals of the black box learning system; states to attain or avoid, conditions to maintain or prevent.

Beyond the specification of the agent-environment coupling, any further information we provide the black box would be considered unnecessary (but, perhaps, useful) knowledge. Clearly, by providing the adaptive system with additional knowledge we can aid the learning process a great deal. However, a black box adaptive system must always be able to, in principle, learn in the absence of such superfluous information. The whole point behind black box learning systems is that we are not required to give them domain-dependent knowledge in order for such a system to learn (beyond simply connecting the system to the task environment).

The specification of the coupling between a black box adaptive agent and an environment characterizes precisely how, in practice, genetic learning systems are actually applied to particular task domains:

- An input language is defined and an input interface mechanism is constructed which encodes information from the environment into the internal language of the machine.
- An output language is defined and an output interface is constructed which interprets the output language, producing the specified behaviors in the environment.
- The payoff values are prescribed for each environmental state. These predefined values are intended to somehow express the degree of benefit or detriment the various environmental states hold for the genetic learning system. Whenever a new state is reached in the environment, the appropriate payoff value is presented to the system as feedback.

Why study black box learning systems? A recent trend in AI research has been the criticism of domain-independent, "weak" methods as being impractical for most interesting problems, favoring instead a more domain-dependent, knowledge intensive approach. However, Holland has claimed that the domain of applicability of weak learning methods may have been underestimated and that it is much larger than is usually believed [Hol86]. Actually, domain-independent (black box) learning algorithms appear to be an important area of research for a number of other reasons as well.

The specification of situating a black box learning algorithm in an environment is an abstract description that encapsulates a general class of machine learning problems. The black box learning problem describes a form of machine learning problem where there is no teacher to intelligently guide the training regimen, or provide the learning mechanism with hints or partial solutions. This class of problems commonly referred to as learning without a teacher, unsupervised learning, learning by experimentation, or automated discovery. The advantage gained in studying such a simply described, abstract problem is that any results or insights achieved will have a wide range of applicability to a number of difficult machine learning problems.

A difficulty with studying knowledge intensive learning methods is that one loses sight of the learning algorithm itself. It is certain that a greater quantity or quality of knowledge will greatly improve the performance of any search process. But, improving the knowledge provided to a learning system tends to obscure the issue of the effectiveness of the learning algorithm itself. Is it adequate? Can it be improved? What are its weaknesses? The black box learning problem focuses directly upon the abstract learning algorithm itself.

A related problem in machine learning research is that we are often unable to effectively compare different learning algorithms. That is, given that we have some particular domain in mind, how can we decide whether one learning algorithm A actually performs better than another algorithm B? The largest obstacle in the way of conducting a fair comparison is the fact that two algorithms may possess differing levels of *a priori* knowledge of the domain. Clearly, by allowing algorithm A to possess more information

than algorithm B initially, we may be simplifying A's problem a great deal relative to B's. Thus, we could not necessarily attribute any advantage in A's performance strictly to the algorithms themselves. There is no feasible way to compare the performance of the two algorithms that begin with different amounts of domain specific knowledge, so we must ensure that the two competing methods begin with *identical* amounts of initial domain specific knowledge. But, this too can become quite difficult once we consider the vastly differing knowledge representation methodologies employed by the different machine learning paradigms. Black box learning algorithms help us in this respect because they require no domain dependent knowledge to begin learning. The simplest way to ensure that two algorithms receive the same amount of domain specific knowledge initially is to provide them both with none (save the coupling information which, of course, would be the same for both). So an interesting advantage of black box learning algorithms is that their performances can be justifiably compared to determine which is actually best for some particular domain or, perhaps, which is superior in general.

A most important issue in machine learning research is the question of how do we get machines to solve problems that we do not already know the answers to (i.e. automated discovery)? Learning machines must eventually deal with situations where we have no background knowledge to provide. Even a learning system that has been given a wealth of domain knowledge at the outset of its task will find, as it adapts to the task domain, that this original knowledge is no longer effective in guiding the learning process [Len83]. Regardless of the approach taken towards constructing learning machines, ultimately, in interesting domains, the machine will have to deal with a black box learning problem in the sense that it will have no *a priori* domain knowledge that continues to effectively aid the search effort.

Learning Through Interaction with an Environment

It has been emphasized that an important feature of black box learning systems is that they must learn strictly through repeated interaction with a task environment. These learning systems behave as autonomous adaptive agents situated in an environment. The nature of the interaction between an environment and agent is specified as follows:

1. The environment provides the agent with a stimulus. (The agent acquires an internal description of the current state of the environment.)
2. The agent responds with some behavior.
3. The environment possibly changes state.
4. The environment may provide the agent with feedback that indicates the benefit (or detriment) to the agent of the current state of the environment.

One particular form of black box learning system is considered throughout this thesis — the genetic learning system. The adaptive behavior of a genetic learning system attempts to capture a number of abstract principles about how learning should occur within this framework.

First of all, it is important to realize that the only measure of success or failure available to the learning system is *feedback* — positive feedback indicating a beneficial situation and negative feedback indicating a detrimental situation. Thus, the goals of an adaptive agent must be to maximize the level of positive feedback attained and minimize the level of negative feedback. To this end, genetic learning systems incorporate a simple policy:

- Avoid repeating actions in states where negative feedback was the result.
- Strengthen the tendency to repeat actions in states where positive feedback was the result.

However, this policy considers only a simplified scenario. What happens if feedback is received only after a long sequence of action has been performed?

- Not only should an adaptive agent appropriately strengthen (or weaken) the action that immediately precedes a feedback event, but it should also strengthen those actions that have "set the stage" for the execution of this final action.

By employing such a policy, genetic learning systems attempt to discover beneficial *sequences* of behavior.

An important feature of genetic learning systems is that, periodically, they will undertake novel courses of action. Sometimes the system may abandon a well established pattern of behavior in an attempt

to discover a more profitable course of action. This prevents the system from becoming "stuck in a rut" where it will perpetually pursue the same mediocre pattern of behavior. This capability also allows genetic learning systems to deal with novel situations. If the system does not know what to do, it will try something.

Whenever a genetic learning system endeavors to undertake a novel course of action, rarely does it attempt an entirely arbitrary action. That is, given some situation, a genetic learning system may try an action that is similar to one that was successful in the same situation, or it may try an action that was successful in a similar situation.

At any one time, there may be a number of actions the learning system may wish to take. Since it is assumed that the system can only perform one action at a time, each of the currently conflicting actions must *compete* for the right to be executed. Through this process of competition, well established, successful actions can be favored without totally ruling out the possibility of attempting novel behaviors. A "survival of the fittest" policy is instituted that removes from consideration the least successful actions and preserves the most successful actions for future competitions. Through this process of continually trying novel patterns of behavior, and throwing out unsuccessful actions, a genetic learning system is constantly attempting to discover and maintain profitable patterns of behavior, thus, improving its performance in the task domain.

Objectives and Overview

The development of genetic learning systems, incorporating a classifier system as the performance component and a genetic algorithm as the adaptive mechanism, constitute a major direction in the development of black box learning techniques. Although the many intriguing ideas and potentials of genetic learning systems have been well written about [HHN86, Hol86], the scope of actual, successful implementations remain characteristically limited to domains such as function/combinatorial optimization or simplified machine learning problems [Gre87b]. Certainly, any domain-independent, "weak" learning method will

experience growing performance difficulties as the scope of the domain is scaled up, but the question still remains: can the performance of genetic learning systems on machine learning problems be improved in a way that does not resort to adding domain-dependent knowledge?

The general objective of this thesis is to attempt to improve upon the basic capabilities of the traditional genetic learning system as a black box adaptive agent. Towards this end, the approach taken is two-fold:

- Through careful examination of some of the existing mechanisms and re-analysis of the fundamental processes.
- Through the construction of computational systems and empirical test.

The first aspect of genetic learning systems to be investigated is the classifier system formalism that comprises the performance system. Upon examination of the representational paradigm underlying these systems a number of subtle weaknesses are exposed, the most serious of these being the limited expressiveness of the representation. These weaknesses and limitations present a number of serious difficulties to the genetic algorithm comprising the adaptive component of the learning system. Unfortunately, completely satisfying solutions to these particular problems still elude genetic algorithm researchers.

The second aspect of genetic learning systems to be investigated is the genetic algorithm itself. The genetic algorithm is constructed from a number of evaluation and selection mechanisms that have been traditionally based upon an economic metaphor. However, a careful examination of these mechanisms reveals a number of weaknesses in the approach. The fundamental selection problem is then reconsidered in its abstract form. From this analysis, new evaluation and selection techniques are suggested that are shown to overcome the weakness of the previous techniques in a series of empirical tests.

Finally, an implementation of a complete genetic learning system is constructed, incorporating the results and suggestions from these investigations. A series of empirical tests are presented, illustrating the performance improvements that are realized with the introduction of the reconstructed mechanisms. A few

other small improvements are investigated and demonstrated as well.

The purpose of this research is *not* to investigate or explore genetic principles for their own sake, in a computational manner or otherwise. Rather, the overall goals and objectives are to construct an effective, efficient black box learning mechanism, regardless of how this may be achieved. The approach taken here merely draws its inspiration from the natural processes of evolution and principles of population genetics. If, for the sake of improving the performance of the learning system, the analogy with natural processes should be violated in some way, so be it.

Before commencing with the investigation and analysis that leads (hopefully) to an improved black box learning system, the structure and operation of genetic algorithms and classifier systems are first introduced and described in some detail.

Chapter 2

An Overview of Classifier Systems and Genetic Algorithms

It had been mentioned previously that two distinct components normally comprise a genetic learning system: the task performance system, a role most often filled by a classifier system, and the learning mechanism, a genetic algorithm. The classifier system is the component that interacts with the task environment: receiving sensory input, responding with some action, and receiving feedback. The genetic algorithm has no direct connection to the external environment but rather it operates strictly by manipulating the structures of the classifier system in an attempt to improve its performance at the domain task. Since it is easiest to understand and explain the behavior of genetic algorithms in terms of their effects upon a performance system, a survey and description of classifier systems is presented first. This is followed by a survey and description of genetic algorithms and how they go about manipulating and modifying the performance characteristics of classifier systems.

2.1. Classifier Systems

Classifier systems are most often used as the performance component of a genetic learning system. However, they are entirely programmable, computationally universal systems in their own right. That is, one can implement arbitrary applications with a classifier system.

Classifier systems are a form of rule-based production system [For81]. A production system typically consists of two main components: a working memory, and a production memory (each production consisting of a condition and action part). The characteristic execution cycle of a production system has three phases:

1. *Match*: Match the conditions of the productions to the elements from the current working memory. Save all successful matches in the *conflict set*.
2. *Conflict Resolution*: Choose one of the matches from the conflict set according to some predefined conflict resolution policy.

3. *Execution*: Execute the action specified by the matched production. This may result in elements being added to or deleted from the working memory, as well as messages printed and external subroutines called.

The structure and behavior of a classifier system is distinct from that of a typical production system in a number of ways. Specifically, classifier systems utilize a modified execution cycle and a radically different representational paradigm. In a classifier system the working memory is called the *message list* and the production memory is called the *classifier base*. A classifier system is comprised of four main components: a message list, a classifier base, an input interface, and an output interface.

Messages

The working memory elements of a classifier system are called *messages*. Messages are represented by strings of fixed length k from the alphabet {0,1}.

Example

A simple conceptualization of a classifier system to play `tic-tac-toe` would represent the entire current state of the `tic-tac-toe` board as a single input message. The contents of the squares can be represented using 2 bits. For example, each of the possible marks `X`, `O` and `B` (blank) could be represented by the patterns: `B = 00, X = 01, O = 10` Thus, an input message would be 18 bit long. (Notice the the pattern `11` has no meaning in this case.) For example, the message `010010000110000000` would denote the following board configuration where the squares are addressed left to right, top to bottom:

X	B	O
B	X	O
B	B	B

Classifiers

The productions of a classifier system are called *classifiers*. Each classifier consists of a condition and action part: `condition` → `action`. Both the condition and action parts are represented by strings of fixed length k from the alphabet `{0,1,#}`. A classifier's condition string can "match" a particular message string under the following conditions:

- When the condition string contains a `0` in position i the message must contain a `0` in position i.
- When the condition string contains a `1` in position i the message must contain a `1` in position i.
- When the condition string contains a `#` in position i the message may contain either a `0` or a `1` in position i. In condition strings the `#` symbol is referred to as a "don't care".

Example

The classifier `#10#` → `##11` would match each of the message strings `0100`, `1100`, `0101`, and `1101`.

Upon execution the only action a classifier may take is to post a message to the message list. The message to be posted is constructed from the classifier's action string as follows:

- When the action string contains a `0` in position i the new message will contain a `0` in position i.
- When the action string contains a `1` in position i the new message will contain a `1` in position i.
- When the action string contains a `#` in position i the new message will contain whatever the message that matched the condition string contained in position i. In action strings the `#` symbol is referred to as a "pass through".

Example

Given the classifier `#10#` → `##11` and the messages `0100`, `1101`, the match with `0100` would produce the new message `0111`, and the match with `1101` would produce the new message `1111`.

Example

Classifiers for a `tic-tac-toe` domain could have a condition string with 18 bits (corresponding to the length of the messages described earlier) to match against board configuration messages. These classifiers also require an action string indicating which square to place a mark into (for simplicity, we will assume that the classifier system always plays `X` and the opponent plays `O`). The first 4 bits of the action string would be the address of the square (the rest of the action string will be ignored). The first 2 bits of the action string specifying the row and the second 2 bits specifying the column, where the squares are addressed as follows:

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

Given such a construction, a classifier which encodes the rule *if every square is blank, then place an* X *in the center square* would be expressed by:

`0000000000000000000` → `0101`.

(Strictly speaking, the condition and action strings are the same length, but for the sake of brevity in the examples, this shorter notation is preferred.)

Environmental Interfaces

The input interface connects the classifier system perceptually to an external environment. The input interface takes stimuli from the environment and constructs the messages that represent those stimuli internally. The output interface connects the classifier system behaviorally to an external environment. The output interface takes those messages from the message list which represent actions to be effected and performs those actions in the external environment.

Extensions

There are a number of extensions to the basic classifier system formalism, which are often utilized in practice. These extensions increase the expressive power of classifiers by introducing negated conditions and multiple conditions.

Classifier systems often include provisions for the negation of condition strings. A classifier with a negated condition string can be executed only if no message exists in the current message list that can match the unnegated form of the condition string. A negated condition string is indicated by prefixing it with a "-" symbol.

Example

Consider the classifier `-#10#` → `0011`. Such a classifier could not be matched in any execution cycle where the message list contained any of the messages `0100`, `0101`, `1100`, or `1101`.

Often classifier systems allow for classifiers that contain multiple conditions. Here, the condition part of a classifier is represented by a list of condition strings, each of which is a (potentially negated) string of fixed length k from the alphabet `{0,1,#}`. Each of the individual condition strings can be matched to a message as described previously. The entire condition of a classifier can be matched only when all of the condition strings making up the condition can be matched. When a multiple conditioned classifier is to be executed, normally the message matching the first unnegated condition string is used by the "pass through" mechanism to construct the new message.

Example

Consider the classifier `0##0,1##1,-#10#` → `##11` and a message list containing only the messages `0110` and `1001`. Here the classifier would be able to match and it would produce the new message `0111`. However, if we were to add the message `0101` to the original messages list, the classifier would no longer be able to match and no new message could be produced.

The execution cycle of a classifier system typically has 5 phases:

1. Post all messages from the input interface to the current message list.
2. Record all possible matches between the classifiers in the classifier base and the messages from the current message list.
3. Select some subset of the recorded matches. For each of these selected matches, generate a message for the new message list.
4. Replace the current message list with the new message list.
5. Process the new message list through the output interface, performing any specified external actions.

There are important differences between the execution cycles of classifier systems and production systems. The classifier system's execution cycle provides for the possibility of executing multiple classifiers in one cycle, in parallel. This possibility is facilitated within the classifier system framework by the limitation imposed upon classifier actions — since the only action classifiers are capable of is to post a message to the new message list, classifier actions do not interfere with one another. Holland makes a strong case for the desirability of what he terms *massive parallelism* as an effective method for representing "mental models" in a rule-based system [HHN86]. Another key difference from production systems is that messages are not persistent from cycle to cycle. In a typical production system the working memory elements can only be removed through an explicit action initiated by the RHS (right hand side) of some production rule (or programmer intervention) [For81]. If certain messages are required to be present over a number of cycles, they must be explicitly and repeatedly reinstated by some particular classifier.

An important feature of classifiers is that their action strings do not necessarily have to encode an external action. If the messages constructed by one classifier match the condition string of another classifier, the two classifiers are said to be *coupled*. It is quite possible to have the output interface ignore many types of messages found on the message list. Thus, long *chains* of classifier executions may be constructed through coupling before any external action is effected.

Holland states that arbitrary finite state machines can be realized by a classifier system and that, furthermore, by making use of coupling and internal recodings that classifier systems are capable of performing recursive operations upon messages. It is claimed that, hence, there are classifier systems which are computationally universal [Hol86].

2.2. Genetic Algorithms

Strictly speaking, classifier systems, by themselves, do not posses any inherent adaptive capabilities. A genetic learning system requires the presence of an adaptive component beyond the basic performance system. This role is filled by a *genetic algorithm*. The structure and behavior of a genetic algorithm can be best understood by describing how it manipulates a classifier system in an attempt to improve its task performance.

As the name suggests, the operation of a genetic algorithm is modeled after some of the processes of population genetics. From this perspective we view the classifier base as a *population* of classifiers, each classifier representing a genotype currently present in the population. Through interaction with an environment, the classifiers will tend to display various levels of *fitness*. Adaptive behavior is then achieved through the application of a reproductive process: the genetic algorithm periodically examines the classifier base, selects "fit" classifiers, and creates offspring by applying *genetic operators* to these parent classifiers. These new classifiers do not replace their parents in the population, but rather, they replace the weak members of the population. The main idea is that through these processes of competition, reproduction and *survival of the fittest* the overall performance of the classifier system can be improved.

Genetic algorithms serve as a heuristic search procedure which is used, in some sense, to optimize the classifier base. The basic algorithm behaves as a stochastic generate and test procedure, generating random classifiers and then testing and evaluating them in the external environment. Thus, the space of possible classifiers is searched stochastically in an attempt to discover the "best set" of classifiers to use.

Genetic algorithms are incorporated into the basic classifier system framework by adding a *genetic*

search cycle to the end of the basic 5-phase execution cycle described previously. The genetic search cycle operates as follows:

1. Choose a genetic operator.
2. Select one or more classifiers from the current classifier base to serve as the parents for the reproductive process. (The exact number of parents to select depends upon the genetic operator that is to be applied.)
3. Apply the genetic operator to the parent classifier(s) to create one or more offspring classifiers.
4. Select a number of classifiers from the current classifier base equal to the number of newly created offspring.
5. Replace these selected classifiers in the population with the new offspring classifiers.

This genetic search cycle may be performed deterministically on every k th cycle, or intermittently on random cycles. Within a particular execution cycle, the genetic search cycle may be executed repeatedly, utilizing various genetic operators.

2.2.1. Genetic Operators

Within the framework of a genetic algorithm the genetic operators serve as the search operators for the classifier space. A wide assortment of different genetic operators appear in a number of different genetic learning systems, however, two genetic operators in particular are the most commonly used: *crossover* and *mutation*.

The crossover operator takes two parent classifiers and constructs two offspring classifiers by swapping contiguous segments of the parents. The crossover operator is the most important of the genetic operators. Through the recombination effects of this operator, genetic algorithms behave more subtly than "random search with preservation of the best" [Hol86]. Contrary to a commonly held view, genetic algorithms operate mainly through a recombination process and less so through mutation processes. An in depth analysis of adaptive behavior under crossover leads one to view successful genotypes (in this case

classifiers) as composed of a number of useful *building blocks*. Through the process of repeatedly applying crossover operations to successful classifiers an implicit, efficient search for optimal combinations of the successful building blocks is being performed [Hol75, Hol86].

Example `#10# → ##11` `0##1 → 11##` *crossover* `#### → 1111` `0101 → ####`

The mutation operator takes a single parent classifier and constructs a single offspring classifier by randomly changing characters at randomly selected positions of the parent. The main importance of the mutation operator is to keep diversity in the population, and to prevent premature convergence to a sub-optimal solution [Bak85].

Example `#10# → ##11` *mutation* `#10# → 0#11`

Another two less common, but important, genetic operators are *inversion* and *cover*.

The inversion operator takes a single parent classifier and constructs a single offspring classifier by reversing a randomly selected, contiguous segment of the parent while maintaining the same meaning of the parent. That is, the semantics of the bit positions move along with the values of the positions. Use of this operator requires that the meanings of the bit positions be explicated and moved accordingly, thus adding a complication to the basic definition and use of a classifier. The idea behind the inversion operator is that it will move co-adapted sets of loci closer together, thus decreasing the likelihood that they would be split up during crossover.

Actually, use of the inversion operator in practice is quite rare. It is thought that inversion is of more use when the classifiers are longer and that their usefulness will grow when larger implementations are attempted [Hol75]. Another view is that, presently, most genetic algorithms converge before the effects of inversion can be felt [Gre87a], and that when more difficult problems are tackled the inversion operator may become important.

Example `#10# → ##11` *inversion* `#10# → #1#1`

The cover operator takes a single message and constructs a random offspring classifier whose condition is guaranteed to match that message. The use of a cover operator arose out of investigations with working systems [Rio86]. Generally speaking, most genetic learning systems begin with a random classifier population. In practice, given non-trivial domains it is quite likely (at least initially) that messages will be encountered that will not match any of the classifiers in the population. When this situation is encountered we are at an impasse, there is no way for the classifier system to respond to the environment. The cover operator is applied in such cases, allowing progress to continue. Normally, this operator is *triggered*, meaning that it only executes when the triggering condition is met.

Example `0101` *cover* `01## → 001#`

Most of these search operators (except cover) are, in reality, incorporating a domain independent search heuristic something like the following: *If some classifier demonstrates good performance then a similar classifier is likely to exhibit good performance as well and, perhaps, it may demonstrate even better performance.*

2.2.2. Selection Processes

Selection processes are an important component of genetic algorithms that arise in a number of different contexts. Specifically, selection processes are utilized for: conflict resolution, classifier reproduction, and classifier extinction.

Conflict Resolution

In the conflict resolution phase of the execution cycle we are faced with the situation of having to choosing one of the successful classifier matches for execution (or possibly a subset of these matches). The simplest selection algorithm utilized for conflict resolution chooses the "best" classifier (or best subset) from the set of matched classifiers [Hol85, Hol86] although, stochastic selection processes are more com-

monly used [De85, Rio86]. The stochastic selection algorithm for conflict resolution chooses one of the matched classifiers (or some subset) at random. Typically, a conflict resolution selection procedure exploits the fact that classifiers exhibit different levels of fitness and generality. The choice is biased by favoring "strong" classifiers with a high level of fitness over "weak" classifiers with a low level of fitness, and by favoring specific classifiers over general classifiers [Hol85, Hol86].

Clearly, we want to select the fittest classifiers in an attempt to demonstrate a high level of performance. Giving a selection bias to stronger classifiers, without totally committing to their selection, as in the stochastic selection procedure, provides us with a mechanism that exploits the immediately useful knowledge expressed by these strong classifiers without preventing the further investigation of untried classifiers. (How this is to be accomplished in an efficient and effective manner is the topic of a later chapter.)

The rationale for giving a selection bias to specific classifiers over more general classifiers is that the more specific a classifier is the more likely it is to be relevant to the situation at hand. The ultimate intent of this bias is for the automatic emergence of default hierarchies, having general "default" classifiers superceded by specific "exception" classifiers in conflict resolution. In fact, the emergence of these types of default structures has been observed in working systems (for example Goldberg's gas pipeline controller described in [HHN86]).

Reproduction

The process of classifier reproduction requires that we select classifiers from the current classifier population to act as the progenitors for a new generation of offspring classifiers. The selection algorithm utilized for reproduction involves a random choice of one or more classifiers from the current population. However, this random choice also is biased, favoring "strong" classifiers over "weak" classifiers. The intention here is clear: by selecting fit parents we are making the broad assumption that the offspring are likely to be fit as well, more so than if we had selected unfit progenitors. The hope is that some of these

offspring will be, perhaps, even more fit than their parents. An extensive mathematical analysis of the adaptive behavior of a classifier population under this reproductive scheme, utilizing the crossover operator is provided in [Hol75] where Holland shows an expected exponential increase in the representation of fit building blocks in the population.

Extinction

In practice, classifier systems are implemented almost universally under the assumption of a fixed size population of classifiers [De87]. Given such a restriction it is clear that for each new classifier we wish to introduce we must remove some other classifier from the current classifier population to make way for its insertion. It is obvious that the selection algorithm we use for this process of classifier extinction should tend to select the "weakest" classifiers from the population, thus instituting a *survival of the fittest* mechanism. That is, we want to be careful not to remove the classifiers that have demonstrated (or may potentially demonstrate) a high level of fitness in the environment. It is important to point out that parent classifiers are not normally replaced in the population by their offspring. Usually, the consequences of removing fit parents from the classifier population are a rapid and severe deterioration of overall classifier system performance. Most often, for this selection procedure, just a simple deterministic choice of the "worst" classifier suffices.

2.2.3. Feedback and Fitness

All of the basic mechanisms comprising a genetic algorithm have been built upon the notion that the classifiers possesses some degree of *fitness* in the environment. An important question is how do we arrive at a measure of a classifier's fitness through observation of its behavior? The answer to this question really depends upon the nature of the machine-environment relationship. However, given the specific such relationship under consideration throughout this thesis (recall discussion about "black box" learning algorithms in chapter 1) the problem becomes, more simply, that of translating environmental feedback into a measure of a classifier's observed fitness.

Holland makes strong use of an economic analogy to tackle this problem [Hol85, Hol86]. In Holland's economic model, a classifier is given a property called its *strength*. A classifier's strength is an estimate of its actual fitness in the environment, larger values indicating a high degree of observed fitness, smaller values indicating a low degree of observed fitness. Upon creation, each classifier is given an initial strength (this initial value can be arrived at in a number of ways: constant initial strength, average of parent's strengths, etc.). These strengths are then modified in two ways: through environmental feedback and through the conflict resolution process.

When the classifier system effects some behavior in the environment, the environment may provide the system with feedback. In Holland's model, feedback takes the form of a non-negative payoff value. This payoff is shared equally amongst all of the classifiers that were executed in the immediately preceding execution cycle, the portions being added to the strengths of the participating classifiers. The basic idea is that a classifier whose action tends to result in large payoffs from the environment will gain a high strength value. This gain in strength will, in turn, increase the likelihood that that classifier will be selected in future conflict resolution phases and, hence, be executed more often. Thus, while the classifier continues to successfully receive payoff from the environment, it will experience a rapid, non-linear growth in strength.

Recall that in the conflict resolution phase of the genetic learning system execution cycle the selection process is supposed to consider both the fitness and generality of the classifiers. Within Holland's framework the strength of a classifier serves as the estimate of its fitness. Holland utilizes a syntactic measure of classifier specificity. The measure of a classifier's specificity is arrived at simply by counting the number of non-# symbols in the condition part. The selection algorithm utilizes a *bidding* procedure:

- Let $S(C,t)$ be the *strength* of classifier C at time t.
- Let $R(C)$ be the *specificity* of classifier C.

  $R(C)$ = the number of non-# symbols in C's condition part.
- Let c be some constant between 0 and 1 (c is called the *bid constant*).

- Then when the condition part of classifier C is matched, it makes a bid:

$$Bid(C,t) = c\, R(C)\, S(C,t)$$

In each execution cycle the highest bidder (or some highest set of bidders) is selected for execution. This bidding formula exhibits the trade-off between strength and specificity discussed earlier.

An important aspect of Holland's bidding scheme is that the winning bidders have their bids subtracted from their strengths. So classifiers lose strength whenever they are executed, but they may regain strength whenever their execution results in payoff from the environment. Thus, those classifiers whose execution results in smaller payoffs, on average, than the bids they pay will diminish in strength ultimately to the point where they may be deleted from the classifier population. A classifier whose execution results, on average, in larger payoffs than the bids paid will gain strength rapidly. This growth in strength results in an increased likelihood that the classifier will be selected as a progenitor for future generations and, hence, the building blocks comprising its structure will increase their representation in the overall population.

2.2.4. The Bucket Brigade Algorithm

Holland has actually considered a number of complications that a learning system may face and the resulting impact upon adaptive behavior. In a classifier system, for instance, many classifiers may operate in combination to achieve a desired effect. Also, given complex environments, the classifier system may not receive environmental feedback after each individual behavior. In such cases, certain classifiers may be valuable to the system because they "set the stage" for other classifiers whose execution results in direct feedback from the environment. For example, in the game of `checkers` the classifier that executes a triple jump will surely receive large amounts of payoff, but what of the previously executed classifiers that created the situation that allowed the triple jump to occur? The problem of deciding which of any early acting classifiers should be given the credit for setting the stage for later, more overtly successful classifiers is known as the *credit assignment problem* [Hol85, Hol86, Sam63]. The *bucket brigade algorithm* addresses this issue by adjusting the strengths of the classifiers as they are activated, improving the estimate of the

classifier's fitness within the contexts in which it is invoked. This algorithm involves only a slight alteration to the bidding procedure outlined previously:

> All classifiers executed in the previous execution cycle who posted a message that was matched by the winning bidders in the present cycle are each paid an equal share of the winning bid [Hol85, Hol86].

The behavior of the bucket brigade algorithm is best explained through an economic analogy: The algorithm behaves as a complex economy operating among the population of classifiers. Each classifier C pays out its strength to its suppliers — those classifiers who send messages to which C may respond. Each classifier C receives payments of strength from its consumers — those classifiers which are enabled by C's messages. Clearly, if a classifier receives more than it pays out, it will acquire a profit and, hence, increase in strength.

Certain classifiers achieve system goals directly and are rewarded by the environment. Any lines of economic exchange within the population which lead to these overtly rewarding behaviors may become ultimately profitable and the classifiers involved will flourish. Any classifiers not belonging to any such profitable lines of exchange will ultimately lose their strength and, perhaps, be replaced in the classifier population.

Thus, the bucket brigade algorithm effectively rewards those classifiers which set the stage for the attainment of overall system goals, while punishing those classifiers which do not.

2.2.5. Performance

The basic principle behind the adaptive capabilities of genetic learning systems is the notion that stronger classifiers encode building blocks that provide for behaviors in the environment that lead to a greater likelihood of successfully achieving the system's goals. Due to competition for execution, reproduction, and survival, the weak classifiers will eventually "die" and, consequently, their deleterious building blocks removed from the population. Strong classifiers will flourish and generate an increasing proportion

of the offspring. Thus, the population will come to contain a higher proportion of successful building blocks. The reproductive process exploits these successful building blocks to rapidly discover the best combinations for constructing successful classifiers. Consequently, the population will come to contain a higher proportion of successful classifiers and, accordingly, the overall performance of the system will improve.

Notice that, in practice, when implementing a genetic learning system (or any adaptive system) for some application environment, the actual payoff values to be given for particular behaviors in particular environmental states must be specified *a priori*. From the classifier system's point of view the feedback it receives from the environment is the direct and *only* measure of its performance. The genetic algorithm, in executing its stochastic search through the classifier space, is attempting to increase the level of payoff it obtains from the environment. Thus, for the process of increasing environmental payoff (the classifier system's point of view) to reflect in corresponding increases in application performance (the system designer's point of view), the feedback mechanisms and payoff values must be formulated such that the level of environmental payoff really does correspond to a reasonable measure of application performance [De85].

2.2.6. External Parameters

The reader may have noticed that the presence of externally settable parameters violates the black box assumptions to some degree. The ability to "tune" the genetic learning system to improve its performance in certain tasks may be considered a feature, however, advantageously setting parameters before learning commences would constitute providing the machine with implicit domain knowledge *a priori*. Genetic algorithms are almost universally implemented with the presence of a number of external parameters. In principle, the parameters could be autonomously adjusted by the learning system itself, thus recovering the black box characteristic. One approach that has been proposed utilizes a (meta) genetic algorithm to adjust the parameters of the original task genetic algorithm in an effort to automatically tune its performance [De85]. However, little has been attempted and even less accomplished in this direction. Another

approach, one that is explored later in this thesis, is to simply develop mechanisms that make use of fewer such parameters.

Chapter 3

Difficulties with Classifier Representations

The first component of a genetic learning system to be investigated is the underlying performance mechanism — the classifier system. Specifically, this chapter is an analysis and critique of the classifier representation paradigm. At a first glance, classifier representations appear to be a simple, effective technique for implementing computational systems that are well suited to manipulation by genetic algorithms. However, upon implementing even simple classifier learning systems, many subtleties arise within the representation which can have a tremendous impact upon the rates and capabilities of adaptation demonstratable by a genetic learning system. The underlying representational methodology imposed by the classifier system framework may possibly be a contributor to the apparent lack of efficacy in substantial machine learning tasks demonstrated by genetic learning systems.

The main focus of the investigation undertaken in this chapter is towards revealing some of the apparent inadequacies of the representational foundations upon which classifier systems are built. This "binary string" representation scheme has proven to be a useful tool for the analysis of adaptive behavior under a stochastic search leading to a number of powerful and general results [BrG87, Hol75]. As a knowledge representation paradigm, however, classifier representations prove to be lacking in expressive power. An important argument will be (in chapter 4) that these inadequacies from a knowledge representation viewpoint necessarily manifest themselves as impairments upon the ability of genetic learning systems to demonstrate effective adaptation to a task environment.

A few miscellaneous concerns will be addressed before presenting a somewhat detailed examination of the expressive properties of the representation.

3.1. Human Factors

To state that programming and debugging with bit strings is a disaster, from a human factors point of view, is an understatement. Classifier systems are clearly negligent in their interface to the user. The need for an adequate environment for programming classifier systems has led to one line of research that has culminated in the development of a front end interface which allows users to program classifier systems using a Pascal-like syntax [Fen88].

3.2. Attribute-Value Representation

As an *attribute-value* representation, classifier systems share in all concomitant weaknesses, particularly with respect to representing *relational* knowledge [De87, HoM87]. In principle, there is nothing that can be represented by a relational ("logical") language that could not as well be represented in an attribute-value language [DiM83, HoM87]. However, doing so could possibly lead to a combinatorial explosion in the number of attributes needed [DiM83]. Thus, in a structural domain, we may be required to encode *very* long classifiers in order to adequately capture domain knowledge.

3.3. Illegal Patterns

The classifier representation encodes knowledge in message strings as a vector of attributes, each attribute containing some particular value from a predefined set of values. For instance, in the `tic-tac-toe` domain messages may encode a board configuration as a vector of the contents of each square, the allowable contents of a square being one of `{X,O,B}`. So, each of these possible marks is encoded as a two bit pattern: `B = 00, X = 01, O = 10`. Notice that not all of the possible two bit patterns were utilized, specifically, the pattern `11` has no denotation assigned to it. We consider unused patterns as *illegal patterns*, because they can never occur in an input message. Clearly, in this type of representation based upon fixed length bit strings we will always have the problem of illegal patterns (unless, of course, we are fortuitous enough to have 2^k possible values for each field). Aside from not seeming to be an elegant property of classifier representations, it can be shown that this property leads to a number of subtle

difficulties when we begin to search the space of possible classifiers.

A classifier that contains an illegal pattern in its condition string will never be executed, because any message that could satisfy the condition would necessarily have this illegal pattern as well, which is impossible. The fact that illegal patterns in classifier conditions are to be avoided presents some difficulties when we use classifier systems.

Normally when a classifier implementation is started out it begins with a *tabula rasa*. That is, the original classifier base provided to a classifier system at the onset of its task is usually a completely random one. As mentioned, illegal patterns in a classifier condition render the classifier useless to the system, so care must be taken during this random generation process to generate strictly legal classifiers. If this is not done, then the responsibility falls upon the genetic algorithm to "weed out" the useless classifiers through its survival of the fittest mechanism.

Another, potentially more serious difficulty arises when we consider searching the space of possible classifiers utilizing a genetic algorithm. Consider the previous representation for the contents of a square: `B = 00, X = 01, O = 10`. Although initially the system may contain classifiers with illegal values, assume that they will have disappeared either by explicit removal or as a result of the survival of the fittest mechanism. Now consider performing random point mutations on classifier strings in a mature population where illegal values have been removed, meaning that the `11` will not exist in any current classifier in the population. Assume also that there is an equal distribution of the three legal patterns throughout the population. (The presence of `#` (don't care) symbols is ignored for clarity in these arguments.)

- Mutating a `00` pattern (`B`) can result in either a `01` pattern (`X`) or a `10` pattern (`O`).
- Mutating a `01` pattern (`X`) can result in either a `11` pattern (illegal) or a `00` pattern (`B`).
- Mutating a `10` pattern (`O`) can result in either a `11` pattern (illegal) or a `00` pattern (`B`).

Thus, given that there is an equal probability of a square containing an `X`, `O`, or `B`, there is a 33% chance of constructing a `B`, a 17% chance of constructing an `X`, a 17% chance of constructing an `O`, and a 33% chance of constructing an illegal pattern. So our search over the space of possible classifiers is biased under

the mutation operator. Not only is there a bias favoring B over X and O, one in three mutations will result in illegal patterns, producing a new classifier that is useless.

Similar difficulties can be encountered under crossover operations as well. Assume, again, that the population contains no illegal patterns, and an equal distribution of Bs, Xs, and Os. Half of the time, a crossover will take place between field boundaries resulting in none of the ill effects encountered under mutation. However, the other half of the time, the crossover operation will take place within field boundaries. Here, similar problems are encountered.

- Crossing a 00 pattern with a 01 pattern can produce a 00 or a 01.
- Crossing a 00 pattern with a 10 pattern can produce a 00 or a 10.
- Crossing a 01 pattern with a 10 pattern can produce a 00 or a 11.

Thus, given that there is an equal probability of a square containing an X, O, or B, there is a 50% chance of constructing a B, a 17% chance of constructing an X, a 17% chance of constructing an O, and a 17% chance of constructing an illegal pattern. So our search over the space of possible classifiers is biased under the crossover operation as well. Notice that the bias favoring B over X and O is even more pronounced under the crossover operation than it was under mutation, and that there still is the possibility of constructing illegal patterns. (Note that although the introduction of the # symbol somewhat reduces search biasing problems, it by no means removes these biasing effects entirely.)

There is an alternative method for dealing with illegal patterns that avoids completely the problem of generating useless classifiers. That is to map illegal patterns redundantly to denote the same value already denoted by some other legal pattern. While ridding us of the problem of generating useless classifiers, this mechanism, again, does not avoid creating search biases.

It is clearly not a desirable property of a search mechanism that it should favor the construction of certain structures over others *a priori*. Unfortunately, such search biases are an inescapable consequence of utilizing the classifier representation paradigm to encode knowledge in a domain. Making use of these biases to our advantage constitutes providing the genetic learning system implicitly with knowledge telling

it where useful structures are more likely to be found, violating the stated black box assumptions.

3.4. Classifier Expressibility

In any rule-learning task, the object of the learning system is to distinguish between those world states where it is desirable to effect a particular action from those where it is undesirable. It is fundamental that we be able to represent a condition for effecting an action which includes all desirable world states and excludes undesirable ones. That this condition be representable within a single classifier, or at least a few classifiers as possible, is a key issue. Clearly, without the ability to in some way encode *generalized* conditions, we would be left with the task of simply enumerating the set of desirable states for each action. Such an exhaustive, table-building approach would constitute a brute force learning algorithm with a prohibitive computational cost even for simple domains. So, generality is a key to computationally efficient learning in that it reduces the number of distinct classifiers the system must discover.

3.4.1. Disjunction

The presence of the "don't care" symbol (#) in classifier representations is important because it allows us to generalize classifiers by expressing *disjunction* in the condition parts of our classifiers. Disjunction allows conditions to match any one of a set of possible values, thus, reducing the number of classifiers required to describe the desirable subsets of environmental states. Despite their obvious usefulness, "don't care"s only provide us with a limited mechanism for representing disjunction. In fact, most of the possible disjunctive combinations are actually unrepresentable in a single classifier.

Consider a domain where we have a field of information that can take on any one of n values. For a classifier representation, typically, we would encode this field with the minimum number of bits possible, which is $k = \lceil \log_2 n \rceil$ bits. Given the three placeholders in conditions {0,1,#}, we can represent $3^k \approx 3^{\log_2 n} \approx n^{\ln 3/\ln 2}$ possible disjunctive combinations of the n values. But there are 2^n possible disjunctive combinations in total. Clearly then, for linearly increasing n there is an exponentially increasing number of unrepresentable disjunctive combinations.

It has been suggested that perhaps by utilizing illegal combinations or adding extra bits to the field this limitation could be overcome. To represent all combinations, we require condition strings of length k, where $3^k \approx 2^n$. Thus, $k \approx \frac{\ln 2}{\ln 3} n$, meaning that to be able to represent every possible disjunctive combination of n values, $O(n)$ bits are required. So if every disjunctive combination is to be representable, we can do little better than to allocate a single bit for each distinct possible value!

The consequences of having unrepresentable disjunctive combinations can range from providing beneficial search biases to pruning any reasonable solutions from the search space. What follows is a rather simple problem from the `tic-tac-toe` domain which demonstrates how the existence of limited disjunction, if not taken properly into account, can effectively remove any possibility of expressing a reasonable solution.

Example

Consider the representation `B = 00, X = 01, O = 10`. Now consider a simple learning task which requires our system to learn how to recognize a full `tic-tac-toe` board. That is, we want our system to learn to respond with a special message, say `1111`, exactly when the board is full (i.e. when no square contains a `B`), no matter how the `X`s and `O`s are arranged. Initially, consider only square 0,0. Here, we simply want to express that whenever square 0,0 contains either an `X` or an `O` (it does not contain a `B`) then we should respond with `1111`. But notice that given the way we have designed our classifier representation it is impossible to represent the condition `X OR O` (equivalently `NOT B`) using `#` symbols. Considering that `X = 01` and `O = 10`, we can see that the only pattern matching both is `##`, but this pattern necessarily matches `B` as well! So with our present representation, the solution to this sub-problem requires two distinct classifiers:

```
01################ → 1111
10################ → 1111
```

Extending this result to consider the whole board requires that we use $2^9 = 512$ classifiers in total (all of which must be discovered by the system)! If we had shown better luck (or insight) we might have

chosen a slightly different representation which would yield a solution requiring *only a single classifier*. Let us alter the representation scheme slightly so that `B = 00, X = 01, O = 11`. Now `X OR O` can be represented by the pattern `#1`, yielding a single classifier which solves the problem:

`#1#1#1#1#1#1#1#1` → `1111`

Obviously this example has been contrived to illustrate the point as blatantly as possible. By changing the bit pattern assignments used, we may dramatically alter the size of the solution set that the system must discover. This difficulty, to be called the *pattern assignment problem* (somewhat analogous to the state assignment problem encountered when designing sequential machines with digital logic), can be stated as follows:

> Given a field that can take on any one of n values, find a bit pattern representation for each value that:
>
> 1. minimizes the number of classifiers needed to express a solution (reduces the size of the solution set), and yet
> 2. minimizes the number of bits used to encode all n values (reduces the size of the search space).

The mere existence of limited disjunction exposes us to the possibility of requiring a solution set which is combinatorially larger than the minimum size attainable by appropriately assigning bit patterns. To avoid such a threat, a classifier system designer is forced to consider the potential effect of any pattern assignment upon the representation of a successful solution, meaning that the designer would have to know the form of successful solutions *a priori*!

Note that some of the classifier system literature includes the provisions for the *negation* of condition strings [Hol86, Rio86]. This additional capability does allow for an effective solution to our particular full board problem in an obvious way. However, this does not solve the problem of unrepresentable disjunctive

combinations in general and, hence, does not rid us of the pattern assignment problem.

Of course, limited disjunction could be thought of as beneficial in some respects. Any way in which we reduce the size of the search space without removing any desirable solution states benefits us by reducing the required search effort.

In conclusion, using a representation scheme which prohibits certain disjunctive combinations from occurring in the search space can be either a blessing or a curse, depending upon whether or not an optimal (or adequate) solution remains representable within a reasonable number of classifiers. However, I would like to emphasize that in using such a representation to our benefit we are, in reality, supplying the learning system with *implicit* domain specific knowledge. That is, we are actually indicating beforehand which combinations are likely to be useful and ruling out others as possible candidates, clearly violating the assumptions we have made regarding the construction of black box learning systems. In any case, if one desires to supply the learning system with domain-dependent knowledge beforehand, it is preferable that such information be *explicitly*, not implicitly, provided.

3.4.2. Position Dependent Semantics

In the simple classifier representation for playing `tic-tac-toe`, we can see that the semantics of the representation are position dependent. That is, to know which particular square a segment of a condition string is referring to, we must know the exact position of the segment in its condition string. In the basic genetic algorithm there is no effective mechanism by which important information can be exchanged between (semantically meaningful) classifier positions. Thus, there is no way for important generalizations to be made across positions in a classifier string. If an important pattern has been discovered for one particular position in a classifier string, this same pattern will have to be *independently discovered for each separate position* where that pattern may be important. This can be illustrated with a simple example from the `tic-tac-toe` domain.

Example

Consider a simple learning task which requires a classifier system to learn how to make legal moves, and suppose that the system has already learned that it is legal to place an `X` in square 0,0 whenever square 0,0 presently contains a `B`. The classifier that encodes this is given by

`00##################` → `0000`

For a complete solution, this information must be generalized to all squares on the `tic-tac-toe` board. But notice that to represent such a generalization the system will require 9 distinct classifiers, one for each square. Notice further, that the presence of the above classifier *in no way* helps the genetic algorithm discover this identical `00` pattern for each of the remaining squares. (If this particular example is not convincing, consider the game of `go`. Here, given an analogous classifier representation 19×19 = 361 distinct classifiers are required just to express what the legal moves are!)

In keeping with the theme of this thesis, it is not wished that this problem be remedied by proposing some new *ad hoc* genetic operator which transfers bit patterns between classifier positions. The belief is that the root of this weakness lies in the way in which we have chosen to represent the domain and not so much within the basic search mechanisms.

So how can the pitfall of position dependent semantics be effectively avoided? A simple maxim would be:

If we want a genetic learning system to be able to effectively generalize its classifiers along some particular dimension of the problem domain, we must ensure that references to instances of that dimension are made explicitly and not through some implicit, position-dependent mapping.

So, in the `tic-tac-toe` domain, we can see that the initial representation scheme was rather naive in this respect (although this was not obvious initially). To provide for the effective generalization of acquired knowledge along the various board positions, all references to board positions must be made explicit in the representation of classifiers. This necessitates a change in the formulation of a classifier system for `tic-tac-toe`.

Example

Instead of having one input message represent the entire board configuration, we will use 9 distinct input messages, one for each square, each message representing the contents of the respective squares. Thus, each message will consist of the two fields: `square` and `contents`, requiring $4 + 2 = 6$ bits per message. Using the previous pattern assignments, the set of messages

```
000001, 000100, 001011

010000, 010101, 011011

100000, 100100, 101000
```

would be used to denote the following board configuration:

X	B	O
B	X	O
B	B	B

Here, we encode an action string with 6 bits, the first 4 bits indicating the address of the square in which to place the mark and the last 2 bits set to `01` indicating that an `X` is to be placed. For example, the classifier `010100` → `010101` says if the center square is empty then place an `X` in the center square. Now we can express the concept of a legal move with the single classifier:

`####00` → `####01`.

By avoiding position dependent references in our representation we greatly enhance the classifier system's capabilities to succinctly express important generalizations. The reader may have realized that even this new representation, while avoiding one pitfall, has introduced a new problem. Let us reconsider the overall problem of playing successful `tic-tac-toe`. Here it is necessary, in order to make appropriate moves, that we consider the contents of more than one square at a time when making decisions. But with the representation just described there is apparently no way for a particular classifier to consider more than a single square at a time. How can we extend the capabilities of the classifiers so that they may consider many squares simultaneously without reintroducing position dependent semantics to the representation? There are two ways in which these types of limitations are overcome in classifier system

implementations: by introducing the possibility of *multiple conditions*, and through coordinated sequences of internal actions (*chaining*). Both of these mechanisms present many new issues relevant to searching the classifier space, and each introduces new problems and limitations. The difficulties associated with chaining will be discussed in detail in the next chapter.

3.4.3. Multiple Conditions and Parameterization

As was mentioned previously, we can extend the basic form of a classifier to include the possibility of an arbitrary number of condition strings. In which case the total condition of a classifier is considered to be satisfied only if each of the condition strings are satisfied (i.e. the total condition is a *conjunction* of the condition strings). Multiple conditioned classifiers are actually common to a number of classifier system implementations [Hol86, Rio86]. Typically, the message that matches the first of the conditions is used by the "pass through" mechanism to construct the action message. With this additional capability we can utilize information about more than one square in a straight-forward manner for the `tic-tac-toe` domain.

Example

Consider the classifier `000001,000101,001011` → `001001`. It says that if square 0,0 and square 0,1 contain `X`s and square 0,2 contains a `B`, then place an `X` into square 0,2 (a good move!). Now consider trying to generalize this particular classifier in a useful way. The reader may have noticed that there was an attempt at some cleverness in the way in which the squares were addressed. By separating the references to rows and columns the hope was to be able to effectively generalize useful information along these dimensions. So an obvious and apparently useful generalization of the previous classifier would be:

`##0001,##0101,##1000` → `##1001`

which is intended to say that no matter what one particular row, if there is an `X` in columns 0 and 1, and a `B` in column 2, then place an `X` into the square in column 2. But, unfortunately, this is not what the classifier actually does. For example, the set of messages `000001, 100101, 101000`

(square 0,0 contains X, square 2,1 contains X, square 2,2 contains B) will satisfy the classifier condition, but the resulting action will not necessarily be a good move. In fact, it is impossible in this formulation to express the general concept of a winning move without resorting to enumerating all of the possibilities, which is, in fact, all that the original formulation provided. So we haven't gained anything with respect to representing the concept of a winning move by introducing multiple conditions.

This particular difficulty is termed the *parameterization problem*, and it arises whenever # (don't care) symbols and multiple condition strings are introduced jointly. The problem is that there exists no mechanism in classifier representations by which we can enforce the equality or inequality of the message bits which match the identical # positions in separate condition strings. Therefore we cannot adequately parameterize solutions which require multiple conditions and some form of agreement between the different conditions about matched messages. This particular difficulty has not been addressed in any of the classifier system literature. So presently, introducing multiple conditions does not provide a general mechanism which allows us to avoid the problems related to position dependent semantics in classifier system implementations.

3.5. Remarks

This chapter has presented and discussed a number of properties of classifier representations that have been deemed to be undesirable features for a performance system that is part of a genetic learning system. It has been established that by utilizing classifier representations we expose ourselves to possibilities of:

- A larger growth in search cost due to position dependent semantics over that which would be possible using more powerful representation schemes. This larger than necessary growth is not avoidable in general, regardless of the amount of domain knowledge we posses.
- A combinatorial explosion in search cost due to the limited capabilities to express disjunction in

classifier conditions. This explosion may be avoidable only if we provide sufficient domain knowledge beforehand.

So, the main point can be summarized as follows: By utilizing simplified representations (and, hence, limiting our capabilities for expressing arbitrary rule forms) we *do not* inherently exclude any problem solutions from being expressible, but we *do* expose ourselves to the possibility of a combinatorial explosion in the number of classifiers required to express these solutions. I would claim further that such an explosion in the solution size necessarily manifests itself as an explosion in the search effort required to discover that entire solution. This creates a problem that we, as classifier system designers, can only avoid with certainty by having sufficient knowledge about the solution *a priori*, clearly violating the stated black box assumptions.

Chapter 4

Discovering Classifier Solutions with Genetic Algorithms

Throughout the preceding chapter a recurrent theme has been to focus on those properties of classifier representations which impair the ability of classifier systems to *succinctly* express solutions to problems. Due to the computational universality of classifier systems, there is always some way in which the representational limitations of individual classifiers can be overcome through the coordinated actions of many classifiers. However, we must keep in mind that the ultimate intent is to automatically discover solutions through some form of genetic search over the space of possible classifiers. However, the simultaneous discovery of multiple classifiers has proven to be quite difficult to achieve using the basic genetic search techniques, this difficulty growing as the required number of classifiers grows.

The effectiveness of a genetic search relies on the fact that a classifier is composed of many subpatterns, called *schemata* [Hol75]. A classifier which has demonstrated a high level of fitness in the environment contains schemata which have individually provided for some aspects of the classifier's generally effective performance. These schemata constitute the *building blocks* from which "optimal" classifiers can be constructed [De85, De87]. Through the reproductive process, useful schemata are continuously recombined in an effort to discover the combination yielding the best performance. However, consider that we have some problem that requires a multiple classifier solution that is expressible only with, say, n classifiers. This implies that each one of the n solution classifiers must contain some important schemata *distinct* from the corresponding schemata of the other classifiers.

Some terminology will be used throughout the discussions to follow. A *sub-solution* refers to one of the n distinct classifiers needed to compose a complete solution to the problem environment. Upon initiating a genetic search through the space of possible classifiers, the original classifier population is random and therefore not likely to contain classifiers that are near any of the sub-solutions. As the genetic search progresses classifiers will begin to appear and be retained in the population that are closer in form to one or the other of the sub-solutions (as a consequence of the survival of the fittest mechanisms). Those classifiers

in the population which are closer in form to some particular sub-solution, more so than to any other, are called the *sub-population* for that sub-solution. So, under a genetic search the classifier base progresses from a heterogeneous population to a collection of specialized sub-populations (optimistically) to a complete collection of sub-solutions. A solution *path* through the classifier space refers to one such progression to a particular sub-solution.

A number of difficulties are encountered when we consider applying a genetic search technique to problems that require many classifiers in order to represent a solution. Specifically, the basic genetic algorithm as described in chapter 2 demonstrates *convergence* and *interference* difficulties in these cases.

4.1. Convergence

Practice has shown that genetic learning systems that utilize the basic genetic search techniques are unlikely to simultaneously converge to a number of distinct, equally useful classifiers. That is, given that a number of distinct classifiers are required to form a complete solution for some problem, the genetic algorithm will tend not to discover and maintain a diverse population simultaneously consisting of classifiers from the different sub-solution paths [GoR87]. The algorithm normally will cluster all of the classifiers about a single sub-solution, even when a number of distinct, equally useful, and necessary sub-solutions exist. This tendency towards homogeneity in classifier populations can cause *premature convergence* to sub-optimal solutions even if an optimal solution is expressible by a single classifier [Bak85].

The convergence phenomenon, called *genetic drift* by some researchers, occurs in finite populations where stochastic errors accumulate to the point where the classifier population ultimately converges to one alternative or another [GoR87]. The cause of this behavior arises from the global competition employed in the classifier extinction process. Recall that whenever a new classifier is created some classifier from the current population must be removed to make way for its insertion. The selection algorithm normally employed in this situation simply removes the weakest classifier from the current population. Initially, the classifier population tends to be diverse, containing classifiers belonging to numerous sub-solution paths.

However, eventually some progress is made down a particular sub-solution path that slightly exceeds that made down other paths. Those classifiers contributing towards progress down this one path will gain higher strengths than other classifiers in the population and, hence, they will acquire a reproductive advantage. Progressively, new classifiers introduced to the population will be the progeny of the dominant sub-population while, concurrently, members of the other sub-populations will gradually be extinguished due to their strength disadvantage. So, the slight initial advantage demonstrated by one particular form of classifier often tends to explode to the point where the population is entirely dominated by that form. This results in the eventual loss of access to the other sub-solution paths as the classifier system converges to a local optimum down the single, dominant path. Unfortunately, there is nothing to guarantee (in finite populations, that is) that the sub-solution path that ultimately dominates the genetic search effort is the most important or effective one to pursue.

There has been some research effort devoted to looking for ways to control and avoid convergence related difficulties. One approach adopted by De Jong is to introduce a *crowding* mechanism to the classifier extinction process (described in [GoR87]). Here, instead of just selecting the worst classifier for replacement, we select some small (2 or 3) subset of the population, replacing the classifier from this subset to which the newly created classifier is most similar. The main idea is that through this replacement strategy classifiers from distinct sub-solution paths will tend not to replace each other in the population and, hence, distinct useful forms will be preserved.

Another approach, taken by Goldberg, has been to have similar classifiers share any feedback received from the environment [GoR87]. This *sharing* mechanism calculates a similarity function between classifiers and distributes payoff amongst the classifiers proportionate to this similarity measure. The key idea behind this approach is that as some sub-population begins to dominate and grow, feedback will be divided amongst more and more classifiers ultimately to the point where the diminishing level of payoff will stablize the number of classifiers in the dominant sub-population.

What these mechanisms share is the idea of somehow mitigating the global competition for survival in such a way as to preserve sub-populations of classifiers that serve distinct purposes. The desire is to dynamically develop sub-populations to exploit different *niches* in the environment wherein the classifiers performing these distinct functions can reside without having to withstand excessive survival pressures exerted upon them by other classifiers serving different purposes.

4.2. Interference

Even considering classifier systems that employ adequate mechanisms for supporting appropriate sub-populations that effectively exploit niches in the environment, there are still difficulties to be encountered in efficiently optimizing each of these individual sub-populations. The problem is that the individual genetic searches progressing down independent sub-solution paths will *interfere* with each other.

The difficulty is, again, one of global competition, here appearing within the reproductive process. Recall that we are assuming a minimum of n distinct classifiers are required to represent the complete solution to some problem domain. As the classifier system begins to exploit the various niches in the environment, gradually improving its performance, stronger classifiers will have been created within the various sub-populations exploiting the different niches. In fact, each of the existing sub-populations will contain a small number of best classifiers. It is not unlikely that the best classifiers from the different sub-populations, while serving distinct functions for the overall system, have come to possess strengths that are comparatively on the same order of magnitude with each other. Thus, strong classifiers from distinct solution paths will often be selected jointly as progenitors for the same offspring because the reproductive selection process disregards the form of the classifiers and considers only their strengths when choosing parents. Such offspring are likely to be worse than either parent because they encode building blocks from necessarily distinct solution paths [Boo85]. That is, two solution paths are distinct precisely because no single classifier can adequately capture the behavior of both paths.

The issue of interfering sub-solutions has been addressed by Booker [Boo85]. He attempts to over-

come these interference difficulties by introducing a *restricted mating policy* to the classifier reproductive process. This procedure allows only those classifiers relevant to the current message list to mate on each invocation of the genetic search cycle. So, for purposes of selecting parents for reproduction, we restrict the set of choices to just those classifiers in the current conflict set. (Actually, Booker also relaxes the notion of a "match" to include degrees of partial matches, thus, conflict sets tend to be larger than under a normal matching scheme [Boo85].) The motivation behind this approach is that only those classifiers that tend to occupy the same functional categories are allowed to mate, meaning that offspring are not created which attempt to solve two disjoint aspects of the problem simultaneously.

In general, the key to limiting this type of interference problem is to mitigate the unbridled, global competition for the right to reproduce. Attempting to achieve dynamic *speciation* within the population depends upon identifying those functional categories of classifiers that are effectively exploiting some particular niche in the task environment. Once identified, some form of restricted mating can be applied within these functional sub-populations. Hence, progress made in one sub-population becomes unlikely to hamper progress in other sub-populations.

Although a great deal of effort has been directed towards finding ways to cope with convergence and interference related difficulties, a completely satisfactory, domain independent solution still remains to be developed. However, even assuming that these particular difficulties may be reasonably avoided, there still remain other, potentially more serious difficulties to be encountered when attempting to discover multiple-classifier solutions. Tasks that require many classifiers to represent a solution specifically pose *independence* problems which must be faced by any search technique that we wish to apply to the classifier space.

4.3. Independence

Recall that we are considering problems that require multiple-classifier solutions that are expressible only with, say, n classifiers. This implies that each one of the n solution classifiers must contain some important schemata distinct from the corresponding schemata of the other classifiers. Thus, at some stage of the overall search, our search efforts for the distinct individuals must diverge and, at this point, progress made towards discovering any one solution classifier will in no way contribute towards finding other solution classifiers (recall the discussion in the section on position dependent semantics). That is, the classifiers which lie along distinct solution paths will no longer provide each other with useful building blocks crucial to continued progress towards their solutions. This means that our overall search efforts must eventually become divided amongst n distinct, *independent* searches.

I would argue that any such increase in the number of classifiers required to represent a solution necessarily manifests itself as a corresponding increase in the overall search effort required to discover the entire solution. So, for instance, a representation that required an exponentially increasing number of classifiers to represent solutions for some class of problems would tend to demonstrate exponential increases in the total search effort required to discover these solutions. This shows clearly why the table building approach discussed previously (making no use of generalization in classifier conditions) is an ineffective learning strategy in that it requires a maximal number of distinct classifiers, which would be reflected in a maximal amount of search effort being required to discover the complete solution.

Unlike the convergence and interference difficulties discussed previously, independence is an inescapable consequence of attempting to simultaneously discover multiple classifiers, regardless of the specific genetic search technique used to search the classifier space.

This would seem to indicate that, in general, the best approach to achieving an effective search of a classifier space is to minimize the number of distinct classifiers required to express complete solutions to problems. This, unfortunately, is not what the classifier representation paradigm guarantees us. As was demonstrated in the previous section, there are a number of characteristics of this representational

formalism that limit the expressiveness of a single classifier. This forces us to compensate by utilizing a number of classifiers to express what might otherwise be representable by a single classifier in a more powerful representation scheme. In fact, there are situations where combinatorial explosions can arise in the number of classifiers that are required as compensation (recall the discussion about limited disjunction and the pattern assignment problem). So, there exists a potential for a corresponding explosion in the amount of search effort required to discover all of the needed classifiers. Having sufficient domain knowledge beforehand is the only way such an unnecessary explosion in the number of classifiers needed can be avoided with certainty, but this clearly violates the desire to utilize classifier systems as an effective black box learning mechanism. Therefore, exposure to the possibility of unnecessary explosions in search time is an inescapable consequence of utilizing classifier representations in black box learning systems.

4.4. Discovering Chains

An often discussed point in the foundational literature on classifier systems is the fact that they are computationally universal [Hol86]. Given this fact one cannot say that there are solutions which exist for some computational problems that a classifier system cannot express. Obviously given any rule-based system that is computationally universal, one can always compensate for limitations in the expressibility of individual rules through the coordinated actions of many rules.

Classifier chaining presents the possibility of demonstrating types of problem solving behavior entirely different from that of the simple stimulus-response that has been assumed throughout most of this thesis. With classifier chaining, problem solving behavior that resembles reasoning, planning, and environmental modeling can be shown to be demonstratable in principle [HHN86]. In fact, the ability to construct chains of classifiers is an essential capability of classifier systems that, in part, provides for their computational universality [Hol86].

However, we must keep in mind that the intention is to automatically discover solutions with some form of mechanized search over the space of possible classifiers, specifically, with a genetic algorithm. As

one might expect, discovering useful chains of classifiers proves to be quite difficult for genetic algorithms. Using the genetic search techniques described previously, one encounters difficulties of *sparse feedback* and *classifier interdependence* when attempting to discover useful chains of classifiers.

4.4.1. Sparse Feedback

Recall that environmental feedback is the only avenue available through which the relative merits of individual classifiers can be evaluated. The more frequent the occurrence of feedback, the more effective the evaluation of the classifiers. A simple illustration of this point can be made by considering the game of `chess`. A system that receives feedback immediately after every move indicating the value of the new position could quickly and effectively evaluate the classifier(s) responsible for that move. However, a classifier system that only received feedback at the end of a game indicating whether it won, lost or drew would have an insurmountably difficult time evaluating the classifiers responsible for, say, the opening few moves. So, the sparsity of the feedback will greatly effect the rate of discovery exhibited by a classifier system, regardless of whether or not internal chains of classifiers are utilized. However, sparse feedback is a more important issue with respect to the presence of internal classifier chains because it is an exaggerated and unavoidable consequence of their use. That is, feedback is received only as an environmental response to an *external* action effected by a classifier system, meaning that environmental feedback cannot be directed specifically towards certain classifiers in some internal chain and not towards others in the same chain.

Why an increasing scarcity of feedback should lead to a less effective evaluation of the classifiers in a population can be technically understood through an examination of the behavior of the bucket brigade algorithm. Recall that the bucket brigade algorithm is the sole mechanism that evaluates classifiers based upon the feedback received from the environment. The bucket brigade behaves in a strictly local manner: currency (strength) is passed only between consecutively executed classifiers [Hol85, Hol86, Rio87]. Consider a useful, recurring internal chain of classifiers. Any direct payoff from the environment would be received directly only by the last classifier in the chain. The next to last classifier in this chain would

observe a resulting gain in strength only on the next execution of the entire chain (recall operation of the bucket brigade). In fact, if we consider a strictly serial, internal chain of n distinct classifiers, it can be seen that n executions of the entire chain are required before the first classifier in the chain realizes its first nominal gain in strength. So clearly, the longer the chain, the more executions are required for the appropriate strengths to propagate down the entire length of the chain and, hence, the longer it takes before an adequate appraisal of the classifiers comprising the chain can be made. These effects have been demonstrated experimentally in [Rio87].

Recently, a technique has been proposed to combat the problem of sluggish propagation through classifier chains by introducing the notion of *bridging classifiers* [Hol85, Hol87, Rio87]. A bridge classifier is activated by the first classifier in a sequence and remains active until the payoff state at the end of the sequence is reached. Such a classifier executes in parallel to the classifiers in the chain and it propagates itself by producing messages that match its own condition as long as the sequence is executing. Since the bridge classifier is present when the payoff is received its strength will be increased the first time the sequence is executed. Thus, this classifier will immediately begin to pay larger bids to all classifiers in the chain. So, the increased strength at the end of the chain is passed immediately to classifiers at the beginning of the sequence [Rio87].

The main idea underlying this approach is to somehow have all classifiers that are responsible for achieving an instance of environmental feedback immediately share in the consequences. That is, after receiving positive feedback we immediately reward all classifiers responsible for bringing about the behaviors that led to its achievement. Correspondingly, after receiving negative feedback we immediately punish all classifiers responsible for bringing about the behaviors that led to the disadvantageous state. Actually, the distinction between internal chains of classifier executions and sequences of classifier executions resulting in external actions is not an important one to make from the point of view of discovering classifier solutions. The sequences of classifier executions that are important to distinguish are those that occur between environmental payoffs. It is within any such sequence that the difficulties associated with

sparse feedback are to be encountered. From this point forward, unless otherwise specified, the term "chain" will refer to any such classifier sequence that occurs strictly between feedback events.

4.4.2. Classifier Interdependence

It is readily apparent that it is far more difficult to discover a solution to some problem that requires a chain of k classifiers as opposed to solving another problem requiring k independent classifiers (chains of length 1). The difficulty encountered when attempting to discover useful chains is the interdependence of the strengths of the classifiers composing them. The strength of any particular classifier in the chain depends a great deal upon which other classifiers are specifically present in the population, meaning that the apparent utilities of the classifiers comprising a chain are interdependent.

Consider a chain of n distinct classifiers that performs some useful function in the environment. If any of the classifiers comprising this chain are missing from the current population, the entire chain cannot be executed and, hence, the final external action responsible for reaching a payoff state will never be effected. Thus, classifiers that form an incomplete chain will lose their bids without recovering strength from the environment, eventually be deleted from the population unless the chain can be completed in time. The performance of a classifier chain depends upon the interaction of all classifiers comprising it. Given an incomplete chain, how is the classifier system to know whether the chain is still too short, or whether there is a weakness in the current chain, or both?

The classifier system is not in a position to independently test individual classifiers but, rather, the classifier system must test *sets* of coupled classifiers. That is, nothing short of a complete chain can demonstrate the merit of the individual classifiers that comprise the chain. Unfortunately, there is a combinatorial explosion in the number of possible sets of coupled classifiers as opposed to the number of possible individual classifiers. So an explosive amount of effort must be expended to test sets of classifiers as compared to testing the same number of classifiers independently.

One situation where these problems might be alleviated somewhat would be the case where the indi-

vidual classifiers participate in a number of different chains. A classifier that has the potential to participate in a number of useful chains is more likely to be successful than a classifier that could potentially belong to only a few useful chains. However, this possibility does not remove the combinatorial explosions that arise when searching the classifier space.

There are currently no effective mechanisms through which useful chains can be efficiently discovered. In fact, in the literature there exists no clear demonstration of useful chains being discovered "from scratch" in classifier populations using genetic search procedures. The arguments in this section related to sparse feedback and classifier interdependence demonstrate why such constructions are exceedingly difficult to achieve in practice.

Perhaps, at this point, it may be useful to reconsider why it was considered desirable to develop chains in the first place. Classifier chaining is essential to demonstrating the computational universality of classifier systems, so, without chains there will exist computational problems that have a solution that a classifier system could not express. Another desirable aspect of classifier chaining is that the computational behavior of the classifier system takes on a more reasoning-like flavor, tending to avoid enumerative table building approaches to problem solving. However, when is a chaining approach more appropriate than a stimulus-response technique? As was discussed previously, it is difficult for the genetic search algorithm to discover solutions requiring large numbers of independent, stimulus-response type classifiers. It has also been demonstrated that is is difficult for these algorithms to discover solutions requiring even short chains of classifiers. Pragmatically speaking, attempting to discover chains of classifiers is appropriate only when the associated search effort can demonstrate a savings over that required to discover a stimulus-response solution.

4.5. Remarks

Holland has stated that the capabilities of parallelism and chaining are the major strengths of the classifier system formalism [Hol86], and this point is not argued. However, by using simplified representation schemes that force the construction of solutions that rely on the coordinated actions of many classifiers (through parallelism or chaining), solutions that might otherwise be expressible with independent classifiers in a more powerful representation, makes the job of discovering these solutions unnecessarily difficult in many cases.

It is perhaps interesting to note that it has been a traditional weakness of genetic algorithms that they do not effectively discover solutions requiring many classifiers and yet this is exactly what the classifier representation paradigm requires (a representation developed specifically for use in genetic systems!). The limitations associated with this representational paradigm necessitate the construction of multiple classifier solutions for a wider range of problems than would be necessary if we considered more powerful representational schemes. This requirement runs contrary to strengths of genetic search as an effective global optimization technique [Bri80].

These weaknesses take on added significance when we consider the fact that classifier representations leave open the possibility for a combinatorial explosion in the number of classifiers needed to express solutions to black box learning problems. This will manifest itself as an explosion in the search effort required which, when the many other weaknesses of the genetic search technique are considered, will prevent a complete solution from ever being discovered for all practical intents and purposes.

Chapter 5

A Reconsideration of Fitness and Selection

This chapter investigates the adaptive component of a genetic learning system — the genetic algorithm. Genetic algorithms have been developed as a domain-independent search technique that is suitable for adapting performance systems that operate in a wide range of environments. The intent here is to attempt to improve upon the capabilities of genetic learning systems without resorting to adding domain-dependent information by re-evaluating and reconstructing some of the basic mechanisms of the genetic algorithm. Specifically, this chapter concerns itself with an examination of the process through which the individual members of a classifier population receive strength — an evaluation of their fitness — based solely upon the feedback obtained by the system. The level of fitness attained by a classifier has a direct bearing upon the likelihood that it is selected in each of the three contexts: conflict resolution, reproduction, and extinction.

This investigation will proceed as follows: First, the various properties of feedback are presented with a discussion of how the different forms impact the capabilities of an adaptive system. The traditional economic fitness mechanism currently employed by most genetic learning systems is discussed, revealing some of the weaknesses of the approach. Next, a reconsideration of the abstract form of the selection problem is undertaken, examining how best to balance the competing needs of exploration and exploitation, followed by a reconstruction of the selection procedure in an attempt to overcome some of the weaknesses of the current approach. And, finally, an account is provided of how fitness is evaluated from feedback under this new selection technique, demonstrating how many of the weaknesses of the economic approach are overcome.

5.1. Feedback

Feedback is a fundamental component required by any adaptive agent. The external environment must provide to the agent some indication of the level of performance the agent is achieving. It is only from this information that an agent can ultimately judge the relative efficacy of its internal computational structures. That is, the feedback it receives is the agent's only perception of its level of performance. Thus, if the measure of feedback increases, the agent necessarily perceives its level of performance as having been increased.

Certainly the nature of feedback available to an agent has as much an impact upon its adaptive capabilities as its perceptual and behavioral capacities. The very nature of the feedback that an environment presents to a learning system will affect the overall adaptive performance that the system can exhibit. These properties may impact the internal architecture of learning systems by restricting the class of appropriate learning algorithms. Feedback models that impose differing requirements can be classified along four orthogonal dimensions: rate, certainty, noise, and dimensionality.

Rate

The rate of feedback refers to the diachronic relationship that exists between the agent's behaviors and the corresponding feedback provided by the environment.

Immediate feedback means that, upon the execution of a single action by the system in the environment, the environment immediately responds with a payoff measure that directly evaluates the fitness of that particular action. Immediate feedback from the environment allows the agent to make swift judgements about the usefulness of the specific actions it has made.

Intermittent feedback means that the environment does not necessarily provide an immediate feedback response that strictly addresses the fitness of the preceding action only. In these cases, feedback may be encountered intermittently, possibly after a long sequence of actions has been taken by the system. It is in this situation where we encounter the *credit assignment problem* [Hol85, Hol86, Sam63]. For example,

in chess we don't know whether we have won or lost the game until (very near) the end of the game. It is trivial to reward the winning move (or correspondingly, to punish the losing move), but what of the previous moves which "set the stage" for these final, overtly successful (detrimental) moves? In order to solve this problem we need some way of assigning the credit or blame when considering a *sequence* of behaviors

Certainty

The certainty of feedback refers to the accuracy of information that the environment provides to the agent.

Certain feedback means that the environment gives a precise indication whether the action (or sequences of actions) was absolutely beneficial or absolutely detrimental. That is, a strictly qualitative assessment is made of the value of the preceding action(s). So, in the case of absolute certainty, there are only two possible feedback values: "good" (True, 1, etc.) or "bad" (False, 0, etc.).

Uncertain feedback means that the environment provides a measure which indicates some degree of success or failure. Uncertain feedback offers a quantitative assessment and is presented typically as a real number. Positive values normally indicating success, negative values indicating failure. The absolute value of a payoff measure represents the degree of success or failure.

In practice, a common difficulty with utilizing uncertain payoff measures is that the values we choose to use are arbitrary. There is no right way to assign these values so the learning system developer often must "pull numbers out of a hat".

Noise

Noise refers to the correctness and consistency of the feedback.

Noiseless feedback means that the feedback is the correct value for the preceding action and environmental state. This means also that whenever a particular action is taken in some particular environmental state, the feedback measure is always the same.

Noisy feedback means that there is no guarantee that the feedback value received is, in fact, the correct one for the particular action and state. The feedback measure provided for a particular action in a particular environmental state may fluctuate on subsequent encounters. A totally noisy environment means that we have essentially no correlation between feedback values and the actual level of performance, nor any consistency between subsequent encounters with identical situations. Clearly in such a scenario, no system could possibly hope to adapt.

Dimensionality

It may be the case that in the domain of interest, there is no single, obvious measure for the level of success. Behavior in the domain may involve consideration of a number of, perhaps non-commensurable, objectives [Sch85]. The dimensionality of feedback refers to the number of independent, distinct ways in which the performance of the agent is judged.

Scalar feedback means that the agent's performance is judged by a single scalar payoff measure.

Vector feedback means that there are a number of distinct considerations to be accounted for in assessing the level of performance achieved by the agent. Here, it is the total responsibility of the agent, not of the environment, to deal with multi-dimensional performance measurements. (Notice that if we impose the responsibility of combining the distinct dimensions into a single measure on the environment we reduce the feedback model to a scalar one.)

For purposes of discussion, this list will be considered an exhaustive characterization of possible models of feedback. Each of the above categories are orthogonal in the sense that any combination of properties, one from each category, for each dimension determines the possible distinct feedback models. Therefore, the basic properties of any particular feedback model are characterized by a tuple `<rate,certainty,noise>` for each dimension. The features in each respective category are ordered in a manner which indicates their relative effects upon learning capabilities, the most desirable features listed first. We can see that from the point of view of constructing an efficient learning system, the best

feedback model possible is `scalar` and `<immediate,certain,noiseless>`. Similarly, the feedback model posing the most difficulty to constructing efficient learning systems is a `vector` of `<intermittent,uncertain,noisy>` feedbacks.

5.2. Economic Fitness

The prototypical mechanism used in classifier systems for translating environmental feedback into a measure of classifier fitness is Holland's economics based method as described in chapter 2. Under this scheme classifiers acquire varying levels of strength depending upon the payoff that has accompanied their demonstrated behaviors. Problems of credit assignment due to intermittent feedback are dealt with by utilizing the bucket brigade algorithm. These fitness mechanisms have provided for the development of rather simple and straight forward selection algorithms whose behavior can be relatively well understood. However, a number of problems exist with these selection algorithms, most of these being artifacts of the underlying economic feedback model.

Proliferation of Parameters

Recall that there are a number of parameters associated with Holland's basic economic model [Hol86] and even more parameters are present in more complicated versions [Hol85]. For instance, the bidding mechanism employed by Holland utilizes an arbitrary bid ratio parameter and may include things like taxation rates, and classifiers must also be given some arbitrary initial strength. Also, the genetic search technique requires the designer to choose a fixed population size and fix the genetic search rates (i.e. how often to apply crossover, or mutation, or inversion, etc.) *a priori*. This proliferation of parameters can become a major difficulty with utilizing any learning system. This becomes apparent when actually testing an implemented classifier system in some problem domain. In particular, the difficulty appears when the system fails to demonstrate effective adaptation to the task environment. The designer is left with little guidance as to what went wrong, which parameters to adjust, how to adjust them, or even if the system is in fact capable of effectively adapting to the environment at all. Is the population too small? Is the bid rate too

high? Are the genetic operators being applied often enough? Or is the problem just too difficult? The only comfort available to classifier system designers is the observation that

> *"There is considerable empirical support for the statement that within reasonable ranges that the values of such parameters are not all that critical."* [De85]

However, changing the parameters can certainly still have an effect. Normally, what the designer must resort to is a repetitive cycle of adjusting some parameter(s) and retrying the classifier system in hopes that effective adaptation may eventually be achieved. This problem is complicated by the fact that the parameter settings all interact in affecting the performance of the learning system. If all of the parameters but one are given appropriate settings, the system may still fail to demonstrate adequate performance. A better understanding of the general learning problem would certainly reflect in a reduction in the number of arbitrary parameters utilized by genetic learning systems.

Ad hoc Payoff and Fitness Measures

An obvious problem with utilizing Holland's economic model in practice is the *ad hoc* nature of the payoff values provided to the classifier system as it performs in some task environment. When a designer chooses a domain in which a classifier system is expected to learn, the payoff values to be associated with particular actions in particular environmental states must be chosen *a priori*. Unlike function optimization problems, in most domains of interest to machine learning researchers it is not readily apparent what numerical values are appropriate to provide as feedback for particular (action,state) pairs. For example, numerically evaluating `chess` positions requires a great deal of domain dependent knowledge on the part of a system designer. How do we deal with domains in which the designer has no previous knowledge? Often what must occur in practice is that the designer chooses intuitive yet arbitrary payoff values. If the classifier system does not display an adequate level of adaptation, these arbitrarily chosen values may be revised. What this means is that the payoff values amount to another parameter that can be adjusted to improve the adaptive performance of the classifier system. This also means that there is one more place to lay the blame if the classifier system fails to perform adequately.

Multidimensional Evaluation

Often the difficulty encountered when choosing appropriate payoff values for some domain is that there are a number of distinct aspects of a system's performance that must be reflected in an evaluation of a classifiers fitness. The most common way to deal with multi-objective feedback is to somehow combine the separate payoff values from the different dimensions into a single scalar value. For example, a linear combination of the payoff values from each dimension. However, the way in which these values are combined is likely to be arbitrary, introducing yet another parameter to the classifier system. Some research has been done for multi-objective function optimization problems that does not combine different dimensions [Sch85] with some promising results in the case where there exists a single genotype which is optimal in all dimensions. (Extending this result to discovering the Pareto-optimal set of genotypes requires the learning system to support multiple sub-populations.)

Strength-Time Dependence

Perhaps the most serious difficulties in using economic based selection algorithms are encountered when we consider the eventual competition between old, somewhat successful classifiers and newly created, potentially better classifiers. This situation clearly demonstrates the tradeoff between *exploration* and *exploitation*. In order to explore new, potentially better forms of classifiers the adaptive system must, temporarily at least, abandon action sequences that already have well established payoff rates [Hol87].

Consider a newly created classifier that is actually better than some other mature classifier that has already acquired a high strength value. Say that this new classifier is also a specialist of the mature classifier. Through the payoff and bidding mechanisms classifiers demonstrate superior strength levels only after a number of executions that have resulted in positive feedback. Clearly then the mature classifier will have acquired a high strength value relative to the remainder of the population. Recall that selection algorithm used for the conflict resolution process deterministically chooses only the winning bidder for execution. So, if the new classifier is not given a sufficient initial strength, it cannot win a bidding competition

against its mature foe and, hence, it will never receive payoff from the environment. This means that the strength of the new classifier cannot grow to the point where it can overtake the mature sub-optimal classifier. Thus, it often occurs that an individual with a relatively high, but sub-optimal, level of fitness is discovered early in the search process. When this happens the selection processes will give it such a strong preference that it can quickly dominate the population and cause premature convergence [De85].

This problem may be avoided to some extent by giving new classifiers high strength values but this approach results in a classifier system that performs poorly for the sake of testing untried, probably poor classifiers. The real issue beneath this problem is the question of how do we allocate trials to optimally balance the competing need of exploring the classifier space with the desire of exploiting previously discovered, apparently useful classifiers? This question is the topic of a forthcoming section.

The Scaling Problem

There are a number of difficulties related to the way absolute strength differences impact the adaptive behavior of the classifier system as opposed to relative strength differences. For instance, late in the search process the classifier population may be legitimately dominated by high strength classifiers which differ on an absolute scale, but these strengths do not necessarily differ on a relative scale. So, it is possible that essentially every classifier contributes equally to future progeny in spite of strength differences. The consequence of this so called *scaling problem* [De85] is that the basic genetic search will be more effective when the classifier strengths differ significantly on a relative rather than absolute scale. It has been expressed by De Jong that a generally effective method remains to be developed that adequately exploits absolute strength differences in spite of relative strength equality [De85].

The Bucket Brigade Algorithm

One final difficulty is one that has been mentioned before, namely the problems regarding the sluggishness of the bucket brigade algorithm. When faced with intermittent feedback from the environment, the classifier system must rely upon the operation of the bucket brigade to effectively evaluate the relative

fitnesses of classifiers in the population. Unfortunately, as environmental feedback becomes sparse, the performance of the bucket brigade deteriorates to the point where an effective estimate of a classifier's fitness takes an unreasonable length of time to achieve [Rio87] (recall the discussion in chapter 4).

In most cases it is not clear how these difficulties are to be avoided given the economic model that is used to arrive at an estimate of a classifier's fitness.

5.3. Selection

The desire to demonstrate the best possible performance generally compels us to consider selecting the best classifier for execution. However, there are likely to be relatively new classifiers in the population that have not been adequately tested to the point where we can assume that they are not better than the observed best classifier. So, we must balance these competing desires of:

- Exploiting the knowledge we already have of the environment (expressed by the classifiers and their strengths) in order to demonstrate what is immediately apparent as the best performance possible.
- Exploring the classifier space in hopes of discovering even better classifiers and consequently achieving superior performance in the long run. This involves departing from the "tried and true" sequences of behavior that may result in poorer performance in the short run.

Balancing the competing needs of exploration and exploitation in an effective manner is the responsibility of the selection procedure. This problem can be seen in its simplest abstract form as the "2-Armed Bandit Problem" [Hol75]. This basic form of the selection problem is introduced and examined with the intent of constructing an efficient and effective selection procedure for use in a genetic learning system.

5.3.1. The 2-Armed Bandit Problem

The 2-Armed Bandit Problem (2ABP) is usually described by the following scenario: You are presented with two *slot machines* unimaginatively called $BANDIT_1$ and $BANDIT_2$. Pulling the arm of one of the slot machines can result in one of two possible outcomes, either the machine will spit out 1 dollar, or it will do nothing. Notice that each pull of a slot machine's arm could be viewed as a *Bernoulli trial*, the event of receiving 1 dollar considered a *success* (it is better than nothing), and the event where nothing happens considered a *failure*. You are told that one of these slot machines has a higher probability of "paying off" on each trial than the other machine, but you are not told which is the one. The problem is then stated:

> Given that you are free to allocate trials in any way you wish, how should the trials be allocated between $BANDIT_1$ and $BANDIT_2$ so that the total expected number of successes is maximized?

The following symbols will be defined for use throughout the following discussions:

- Let p_i be the true probability that $BANDIT_i$ will actually payoff on a given trial.
- Let t be the total number of trials attempted.
- Let $n_i(t)$ be the number of trials attempted on $BANDIT_i$ and let $k_i(t)$ be the number of observed successes up to time t. So $t = n_1(t) + n_2(t)$.
- Let $\hat{p}_i(t)$ be the proportion of successes observed on $BANDIT_i$ up to time t. It is worthwhile to notice that this observed proportion of success $\hat{p}_i = \frac{k_i}{n_i}$ is the maximum likelihood estimator for p_i.

Note that it is an underlying assumption that all trials are independent of one another and that each trial terminates. It is also assumed that the payoff probabilities of the slot machines remain constant throughout the experimentation.

Of course, instead of considering slot machines, the 2ABP could also be expressed in terms of classifiers: You are given two classifiers that achieve environmental feedback at different rates. How are the trials to be allocated so that the total expected amount of feedback accumulated is maximized?

Asymptotic Optimality

What is the best strategy for allocating trials to the two slot machines? If we knew, for instance, that $p_1 > p_2$ then, clearly, the best strategy would be to play $BANDIT_1$ exclusively. With this strategy, the total expected number of successes would be maximized. The problem of optimally allocating trials when the probabilities p_1 and p_2 are not known beforehand is the main topic of investigation of this section.

One sense of optimality that is important to consider here is the notion of *asymptotic optimality*. A procedure is said to be asymptotically optimal if it guarantees that the observed proportion of successes converges to $p = max(p_1, p_2)$ when t, the total number of trials, approaches infinity. That is, as the number of trials we perform grows without bound, the overall performance of an asymptotically optimal procedure will be such that we could have done no better even if we had known the identity of the superior bandit *a priori*.

Throughout most of the discussions to follow it will be assumed, without loss of generality, that $p_1 > p_2$, but this (obviously) will not be known by the procedures beforehand.

Robbins' Algorithm

Actually, an asymptotically optimal solution to the 2ABP has existed since 1952 when Robbins introduced a simple, but artificial procedure [Rob52].

Let $\alpha = \{1 < a_2 < a_3 < \cdots\}$ and $\beta = \{2 < b_2 < b_3 < \cdots\}$ be two disjoint, unbounded sequences of positive integers such that the proportion of integers $1, 2, 3, \ldots, n$ which are either elements of α or β tends to 0 as $n \to \infty$. Now suppose that after t trials, k_i successes have been observed in the n_i trials associated with $BANDIT_i$. On the next trial, $t+1$, we test the bandit with the highest observed proportion of success unless $t+1 \in \alpha \cup \beta$. If $t+1$ happens to be a member of one of these prescribed sequences, then the next trial must test $BANDIT_1$ if $t+1 \in \alpha$, or test $BANDIT_2$ if $t+1 \in \beta$.

The asymptotic optimality of this procedure can be simply demonstrated by first noticing that there are an unbounded number of integers in the sequences α and β, meaning that each bandit will receive an

infinite number of trials.

$$\lim_{t \to \infty} n_i(t) \to \infty$$

One important property of the estimator $\hat{p}_i$ is its *consistency*, meaning that $\lim_{n \to \infty} \hat{p}_i = p_i$ [LaM81, MSW81]. This means that there must exist a T such that for all $t \geq T$, $\hat{p}_1(t) > \hat{p}_2(t)$. So, after time T, as $t \to \infty$, the procedure will choose to test only $BANDIT_1$ except at those prescribed times $t \in \beta$ when $BANDIT_2$ will be tested. (Notice that we cannot place a finite bound on the value of T, but there still exists an unbounded number of integers t such that $t \geq T$.) But, by definition, the proportion of integers 1, 2, 3,...,n that are elements of β tends to 0 as $n \to \infty$. So it follows from the laws of large numbers that

$$\lim_{t \to \infty} \frac{n_1(t)}{t} = 1, \lim_{t \to \infty} \frac{n_2(t)}{t} = 0$$

and, hence, that

$$\lim_{t \to \infty} \frac{k_1(t) + k_2(t)}{n_1(t) + n_2(t)} = p_1 = max(p_1, p_2)$$

[Bat80, Rob52].

An unbounded number of trials will be performed on the inferior $BANDIT_2$, but the number of such tests will be sufficiently sparse so as to leave the asymptotic proportion of tests unaffected as though only $BANDIT_1$ were used throughout [Kum85].

Unfortunately, this procedure is not that useful from the point of view of genetic learning systems. It is not clear how procedures that require sparse and unbounded number sequences can be usefully applied, especially when we consider extending this construction to an undetermined number of competing classifiers that may not all be introduced simultaneously.

There have since been a number of algorithms constructed that also demonstrate asymptotic optimality, some of them being simple variations of Robbins' algorithm [Kum85], and others that introduce statistical procedures to help decide which bandit to choose on each trial [LaR85, Pol78]. All of these procedures are *deterministic* in that for each trial they will specify one particular bandit to be tested on the next trial.

Another more recent approach taken by some researchers has been to utilize a *stochastic* selection procedure for choosing a bandit to test on the next trial [Bat80]. The stochastic approach offers more flexibility for the ways in which the 2ABP may be dealt with and, consequently, it is the method that will be developed here.

5.3.2. Asymptotically Optimal Stochastic Procedures

In this section, I will develop and characterize a class of stochastic procedures for the 2ABP. From the stated assumptions that characterize this class, it will be shown that any procedure that satisfies the conditions of class membership is an asymptotically optimal one.

Weighted Stochastic Selection

A stochastic sampling procedure for the 2ABP can be characterized as follows: Each of the two bandits are given *weights*, w_1 is given to $BANDIT_1$ and w_2 is given to $BANDIT_2$. Let us place the following constraints on the values of the weights:

$$w_1 + w_2 = 1,$$

$$0 \leq w_1, w_2 \leq 1.$$

Let Θ be a continuous random variable with a uniform density over the interval [0,1]. Then a simple weighted stochastic selection procedure can be performed as follows: Let θ be a random sample from the uniform distribution. One of the alternative bandits is selected at each stage by the following rule:

- If $0 \leq \theta \leq w_1$ then $BANDIT_1$ is selected.
- If $w_1 < \theta \leq 1$ then $BANDIT_2$ is selected.

Effectively, the weight w_i of a bandit is the probability that the bandit will be selected in a particular trial.

Of course, for purposes of solving the 2ABP the weights w_1 and w_2 will be dynamically modified, somehow according to the observed behavior of the bandits. If we assume that the update policy for modi-

fying the weights converges in a manner such that

$$\lim_{t \to \infty} w_i(t) = \text{some constant } c_i$$

then it is clear that

$$\lim_{t \to \infty} \frac{n_i(t)}{t} = c_i .$$

That is, if the update policy for modifying the weights converges asymptotically, then the overall proportion of trials devoted to any particular bandit must also converge. It will be seen that this asymptotic behavior under the condition of converging weights is an important property of stochastic selection procedures for the 2ABP.

A Class of Asymptotically Optimal Stochastic Algorithms for the 2ABP

A family of asymptotically optimal procedures for the 2ABP can be easily constructed given the stochastic selection mechanism described previously.

Assumptions

At each stage t of the procedure, let the weight $w_i(t)$ for each bandit be assigned by a function $OP_{ij}(t)$ (where j is the other bandit). Let us further assume the following properties of $OP_{ij}(t)$:

(i) $0 < OP_{ij}(t) < 1$ for finite t.

(ii) $OP_{ij}(t) = 1 - OP_{ji}(t)$.

(iii) If p_1 is truly greater than p_2 then $\lim_{t \to \infty} OP_{12}(t) = 1$, and $\lim_{t \to \infty} OP_{21}(t) = 0$.

Asymptotic Optimality

As we have seen, given that $p_1 > p_2$, all that is really required to show the asymptotic optimality of any algorithm for the 2ABP is that

$$\lim_{t \to \infty} \frac{n_1(t)}{t} = 1, \text{ and } \lim_{t \to \infty} \frac{n_2(t)}{t} = 0.$$

That is, the behavior of the procedure is such that, in the long run, we could not improve on its performance even if we knew the identity of the superior bandit *a priori*.

Lemma

If the weights are assigned at each stage of the procedure by a function with the properties of OP then $\lim_{t\to\infty} n_i(t) \to \infty$ (each bandit receives an unbounded number of trials).

Proof

Assume that there exists some *finite* integer T such that after trial T $BANDIT_i$ is never again tested. This implies, beginning at the next trial (trial $T+1$), that $w_i = 0$, and that w_i continues to be 0 thereafter.

But $w_i(t) = OP_{ij}(t)$, and under our assumptions about the behavior of OP, $OP_{ij}(T+1) = 0$ implies an infinite number of trials, a contradiction of assumption (1).

Theorem

If p_1 is truly greater than p_2, and the weights are assigned at each stage of the trial by a function with the properties of OP, then

$$\lim_{t\to\infty} \frac{n_1(t)}{t} = 1, \text{ and } \lim_{t\to\infty} \frac{n_2(t)}{t} = 0.$$

Proof

Recall from the discussion on the behavior of the stochastic selection algorithm that $\lim_{t\to\infty} \frac{n_i(t)}{t} = \lim_{t\to\infty} w_i(t)$ which, in this case, $= OP_{ij}(t)$.

From the lemma we have that each bandit receives an unbounded number of trials.

So, under our assumptions about the behavior of OP, if p_1 is truly greater than p_2 we have that

$$\lim_{t\to\infty} OP_{12}(t) = 1, \text{ and } \lim_{t\to\infty} OP_{21}(t) = 0.$$

which yields the result.

5.3.3. Convergence and Finite Optimality

It turns out that constructing asymptotically optimal procedures for the 2ABP is not all that difficult. Nor are asymptotically optimal algorithms necessarily interesting from a practical point of view in that there are algorithms that yield optimal performance in the limit but are *arbitrarily slow to converge*. A useful and complete solution to the 2ABP would require the construction of an asymptotically optimal procedure that also exhibits an optimal rate of convergence. Some steps in this direction have been made by considering the 2ABP under the restriction of a finite number of trials. For example, Sato *et al* have constructed a deterministic procedure that maximizes the expected number of successes given that the two probabilities p_i, p_j, and the number of trials N are all known beforehand [SAT84]. The algorithm splits the trials into two phases: the testing phase consisting of the first n^* trials, and the exploitation phase which strictly tests the bandit that achieved the highest observed proportion of success in the test phase. (This construction is very similar to the treatment given in [Hol75].) Clearly, this construction is not general enough to be of use as a selection mechanism in adaptive classifier systems. A useful procedure is one that can exhibit fast convergence without assuming knowledge about the values of the probabilities and a fixed number of trials.

5.3.4. A Fast Algorithm for the 2ABP

It has been shown that a stochastic, asymptotically optimal algorithm can be constructed from any function that satisfies the three specified properties of OP. The approach taken here is to construct a function that satisfies these crucial properties of OP and yet adjusts the weights in an appropriate manner so as to achieve a high rate of convergence. Consider the following function:

$$PP_{ij}(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_{ij}(t)} e^{-\frac{1}{2}x^2} dx,$$

where

$$z_{ij}(t) = \frac{\hat{p}_i - \hat{p}_j}{\left[\frac{\hat{p}_i \hat{q}_i}{n_i} + \frac{\hat{p}_j \hat{q}_j}{n_j} \right]^{\frac{1}{2}}}$$

and

$$\hat{p}_i = \frac{k_i}{n_i},$$

$\hat{q}_i = 1 - \hat{p}_i$.

(Note that $z_{ij}(t)$ is defined only in the region $n_i > 0$, $n_j > 0$, and, for our purposes, both $0 < k_i < n_i$ and $0 < k_j < n_j$.)

Theorem

In the region where it is defined, the function PP satisfies the three properties of OP and, hence, yields in an asymptotically optimal algorithm for the 2ABP.

(i) $0 < PP_{ij}(t) < 1$ for finite t.

(ii) $PP_{ij}(t) = 1 - PP_{ji}(t)$.

(iii) If p_1 is truly greater than p_2 then $\lim_{t \to \infty} PP_{12}(t) = 1$, and $\lim_{t \to \infty} PP_{21}(t) = 0$.

Proof of (i)

$PP_{ij}(t) = 0$ implies that $z_{ij}(t) = -\infty$, and

$PP_{ij}(t) = 1$ implies that $z_{ij}(t) = +\infty$ [LaM81].

But, no denominator of z vanishes in the defined region, thus, $z(t)$ remains finite for finite t.

Proof of (ii)

It is a well known property of the integral

$$F(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-x^2/2} dx$$

that for values z_1, z_2, where $z_1 = -z_2$ that $F(z_1) = 1 - F(z_2)$ [LaM81, MSW81].

By inspecting the formula for z it is clear that $z_{ij}(t) = -z_{ji}(t)$.

Thus, $PP_{ij}(t) = 1 - PP_{ji}(t)$.

Proof of (iii)

Recall that $0 < PP_{ij}(t) < 1$ for finite t, and that this property of PP implies $\lim_{t\to\infty} n_i(t) \to \infty$, and $\lim_{t\to\infty} n_j(t) \to \infty$.

Consider the denominator of the formula for $z_{ij}(t)$ which can be written

$$\left[\frac{\hat{p}_i \hat{q}_i}{n_i} + \frac{\hat{p}_j \hat{q}_j}{n_j}\right]^{\wedge}$$

Clearly, this quantity vanishes in the limit as both $n_i(t) \to \infty$ and $n_j(t) \to \infty$. So, $\lim_{t\to\infty} z_{ij}(t) \to -\infty$ or $+\infty$, because the numerator of z does not vanish in the defined region.

Consider the numerator of the formula for $z_{ij}(t)$: $\hat{p}_i - \hat{p}_j$. It has been previously mentioned that $\lim_{t\to\infty} \hat{p}_i = p_i$, the true value of the probability of success of *BANDIT*$_i$ [Bat80, LaM81, MSW81]. So, the numerator converges in the limit: $\lim_{t\to\infty} \hat{p}_i - \hat{p}_j = p_i - p_j$.

Given the fact that $p_1 > p_2$, the numerator of $z_{12}(t)$ must become positive in the limit as $t \to \infty$. Correspondingly, the numerator of $z_{21}(t)$ must converge to a negative value.

Thus, $\lim_{t\to\infty} z_{12}(t) \to +\infty$, and $\lim_{t\to\infty} z_{21}(t) \to -\infty$.

This, of course, means $\lim_{t\to\infty} PP_{12}(t) = 1$, and $\lim_{t\to\infty} PP_{21}(t) = 0$.

Derivation of the Algorithm

It has been mentioned that a number of researchers have utilized statistical techniques in order to make deterministic decisions about which bandit to test on any particular trial [LaR85, Pol78]. The reader has probably guessed by now that the function PP was not chosen in an entirely arbitrary fashion. Actually, the function $PP_{12}(t)$ is intended to express the "degree of certainty" that $p_1 > p_2$, given the observations made up to time t.

How can we express this "degree of certainty" that p_1 is, in fact, larger than p_2 given the observations $\hat{p}_1$ and $\hat{p}_2$? For instance, it is nonsensical to use a probability like $P[p_1 > p_2]$ because neither p_1 nor p_2 are

random quantities. This probability is either 0 or 1 depending upon whether or not p_1 really is greater than p_2.

However, $\hat{p}_1$ and $\hat{p}_2$ are random observations. That is, these estimates, being calculated from a single sample, will vary in a random manner from sample to sample. So, our notion of "degree of certainty" must be based somehow upon the relationship between the estimators and the target parameters.

Consider the problem of expressing the "degree of certainty" that some target parameter θ is truly greater than some constant c, given only an observed estimate $\hat{\theta}$. First, consider constructing a *one-sided confidence interval* for θ.

- Let $\hat{\Theta}$ be an estimator for the target parameter, where
- $E(\hat{\Theta}) = \theta$, and
- $\sigma^2_{\hat{\Theta}}$ denotes the variance of $\hat{\Theta}$.

Notice that the pivotal quantity $Z = \dfrac{\hat{\Theta} - \theta}{\sigma_{\hat{\Theta}}}$ has an approximate standard normal distribution $N(0,1)$ [MSW81]. This means that we can determine the (approximate) probability that the random variable Z will be within some interval.

$$P[z_\alpha > Z] = 1-\alpha,$$

$$P\left[z_\alpha > \frac{\hat{\Theta} - \theta}{\sigma_{\hat{\Theta}}}\right] = 1-\alpha,$$

$$P[\hat{\Theta} - z_\alpha \sigma_{\hat{\Theta}} < \theta] = 1-\alpha$$

[MSW81].

So, the probability that the target parameter θ is not more than z_α standard deviations smaller than $\hat{\Theta}$ is $1-\alpha$.

The relationship between $1-\alpha$ and z_α is given by the following formula:

$$1-\alpha = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_\alpha} e^{-\frac{1}{2}x^2} dx.$$

So, by specifying α, the confidence coefficient, we can determine z_α, yielding an interval that has a $100(1-\alpha)\%$ chance of containing θ. Conversely, we can first specify z_α which, in turn, can be used to determine $1-\alpha$ [LaM81, MSW81].

Consider an alternative, but equivalent form of the previous probability statement.

$$P[\theta - z_\alpha \sigma_{\hat{\theta}} < \hat{\Theta}] = 1-\alpha.$$

The probability that the value of $\hat{\Theta}$ is not more than z_α standard deviations smaller than the target parameter is also $1-\alpha$. With this formulation we can make statements about the probability that $\hat{\Theta}$ is within a certain interval, provided that we know θ.

$$P[c < \hat{\Theta}] = 1-\alpha.$$

The probability that $\hat{\Theta}$ would be greater than c is $1-\alpha$. So, we have $c = \theta - z_\alpha \sigma_{\hat{\theta}}$, which implies that

$$z_\alpha = \frac{\theta - c}{\sigma_{\hat{\theta}}},$$

thus determining $1-\alpha$. However, we cannot calculate z_α (and, hence, we cannot calculate the probability $1-\alpha$) because θ is unknown.

Recall that the intent is to somehow measure the "degree of certainty" that the target parameter θ is truly greater than some constant c, given only an observed estimate $\hat{\theta}$. Clearly, if $\hat{\theta}$ is much greater than c we would be more certain that $\theta > c$ than we would be if $\hat{\theta}$ were closer to c. Similarly, if $\hat{\theta}$ is based upon a large number of trials we would be more certain that $\theta > c$ than we would be if the estimate were based upon fewer trials. One such measure that captures these intuitions is the following:

> If the observed estimate $\hat{\theta}$ is assumed to accurately reflect the true value of the target parameter θ, what would be the probability, upon repeating the experiment, that we would observe estimates $\hat{\Theta}$ that are greater than c?

For our purposes here, the "degree of certainty" that $\theta > c$ given only the observed estimate $\hat{\theta}$, denoted $C[\theta > c \mid \hat{\theta}]$, will be defined to be this measure — the probability that repeating the experiment would yield $\hat{\Theta} > c$, under the assumption that $\theta = \hat{\theta}$.

$$C[\theta > c \mid \hat{\theta}] = P[\hat{\Theta} > c \text{ assuming that } \theta = \hat{\theta}] = 1-\alpha.$$

So, $1-\alpha$ can be determined by using $z_\alpha = \dfrac{\hat{\theta} - c}{\sigma_{\hat{\theta}}}$.

Using this particular quantity to serve as a measure of the "degree of certainty" about the true value of the target parameter is not fully justified in any formal, mathematical sense. Perhaps there may be better ways to capture the notion of "certainty". However, the measure used here can be considered a reasonable one to use in that it captures many of the intuitions we have about the way certainty should behave. The assumption that the observed estimate is near enough to the target parameter is appropriate one to make if the estimator is based upon a large number of trials.

Recall the original problem of measuring the "degree of certainty" that $p_1 > p_2$, given observed estimates $\hat{p}_1$ and $\hat{p}_2$. First of all, notice that $p_1 > p_2$ implies that $p_1 - p_2 > 0$. Notice also that the estimator $\hat{P}_1 - \hat{P}_2$ is an unbiased and consistent estimator for this quantity [LaM81, MSW81].

- $E(\hat{P}_1 - \hat{P}_2) = p_1 - p_2$.
- $Var(\hat{P}_1 - \hat{P}_2) = \dfrac{p_1 q_1}{n_1} + \dfrac{p_2 q_2}{n_2}$.
- The pivotal quantity $Z = \dfrac{\hat{\Theta} - \theta}{\sigma_{\hat{\theta}}}$, where $\theta = p_1 - p_2$, and $\hat{\Theta} = \hat{P}_1 - \hat{P}_2$ has an approximate standard normal distribution $N(0,1)$ [MSW81].

So, the previous technique can be applied in this case to arrive at a measure of the "degree of certainty" that $p_1 > p_2$. The "degree of certainty" that $p_1 > p_2$, given the observed estimates $\hat{p}_1, \hat{p}_2$, will be defined to be the probability that repetition of the experiment would yield $\hat{P}_1 - \hat{P}_2 > 0$ under the assumption that $p_1 = \hat{p}_1$ and $p_2 = \hat{p}_2$.

$$C[p_1 > p_2 \mid \hat{p}_1, \hat{p}_2] = P[\hat{P}_1 > \hat{P}_2 \text{ assuming that } p_1 = \hat{p}_1 \text{ and } p_2 = \hat{p}_2] = 1-\alpha.$$

This probability is determined by the formula

$$1-\alpha = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z_\alpha} e^{-\frac{1}{2}x^2} dx,$$

where

$$z_\alpha = \frac{\hat{p}_1 - \hat{p}_2}{\left[\frac{\hat{p}_1\hat{q}_1}{n_1} + \frac{\hat{p}_2\hat{q}_2}{n_2}\right]^{\frac{1}{2}}}.$$

Notice that this is exactly the formula for z_{12}, thus, yielding the result that

$$PP_{12} = C[p_1 > p_2 \mid \hat{p}_1, \hat{p}_2] = 1 - \alpha.$$

The selection procedure that utilizes this function PP shall be referred to as a "likelihood-based" selection technique. The behavior of this algorithm is explored throughout the rest of this thesis.

Convergence

The behavior of the likelihood-based selection technique is intuitive and simple to understand. At each stage of the trial the bandits are given weights that correspond to the "degree of certainty" that their success rate is actually better than their rival's, given the previous observations.

If one bandit's observed payoff proportion is greater than the other bandit's, this apparently superior bandit will receive a higher weight than the inferior bandit and, thus, it will become more likely to be tested in future trials. If the observed advantage is slight, the difference in weights is smaller than it would have been if the observed advantage was large.

Another important aspect of the behavior of this algorithm is that as the amount of acquired information grows, so does the "degree of certainty" that one bandit or the other possesses a higher success rate. This increase in certainty is reflected directly in an increase in the weight advantage given to the apparently superior bandit. So, even a slight advantage in observed payoff rate will result in an exaggerated weight difference as more trials are observed.

It has been shown that if one bandit possesses a truly superior success rate the algorithm will discover this fact and exploit it to the extent that, in the long run, it will allocate trials in an optimal manner. More importantly, the stochastic algorithm that uses the function PP to assign weights at each stage of the exper-

iment converges quickly to correct decisions in an apparently efficient manner.

Experiment

These intuitions appear to be supported by the empirical evidence in that the algorithm does converge to correct decisions in rapid fashion. The likelihood-based selection technique demonstrates a much higher rate of convergence than a traditional, economics based selection mechanism like the one described in chapter 2. To demonstrate this point, experimental results were obtained to compare the two approaches. Consider the graphs shown in Figure 5.1 and Figure 5.2. Both graphs show the results of an experiment with two one-armed bandits, one bandit A with a payoff proportion of 3/4, and the other bandit B with a payoff proportion of 1/4 (details of the procedures followed in the 2ABP experiments can be found in Appendix A1). Figure 5.1 shows the resulting proportion of trials allocated by the economic selection technique and Figure 5.2 shows the resulting proportions allocated by the likelihood-based selection technique. It is clear upon examining these graphs that the likelihood-based technique converges to the correct proportion — 100% A, 0% B — at a much faster rate than the economic technique.

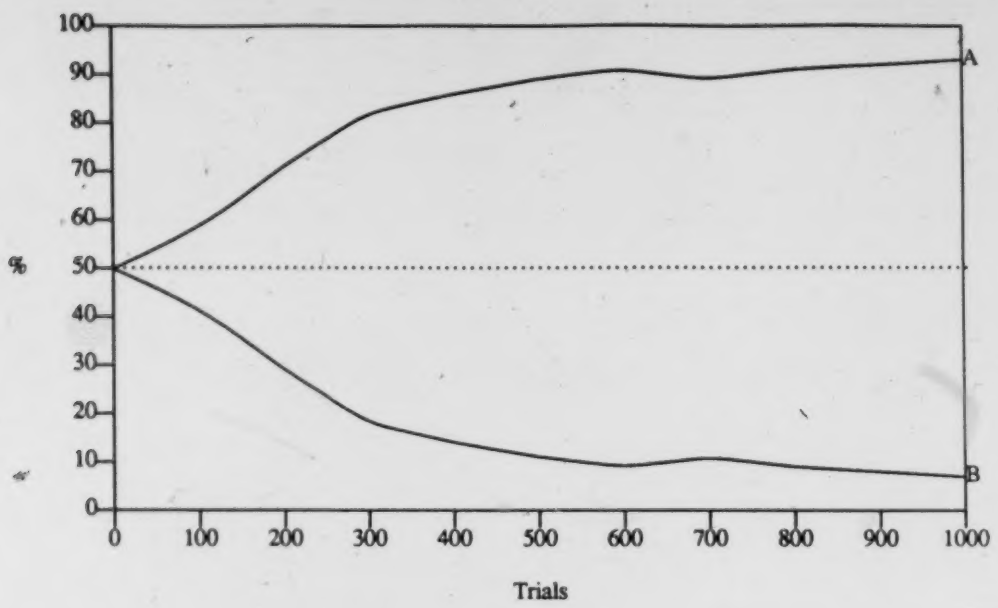

Figure 5.1 2ABP Experiment with Economic Selection

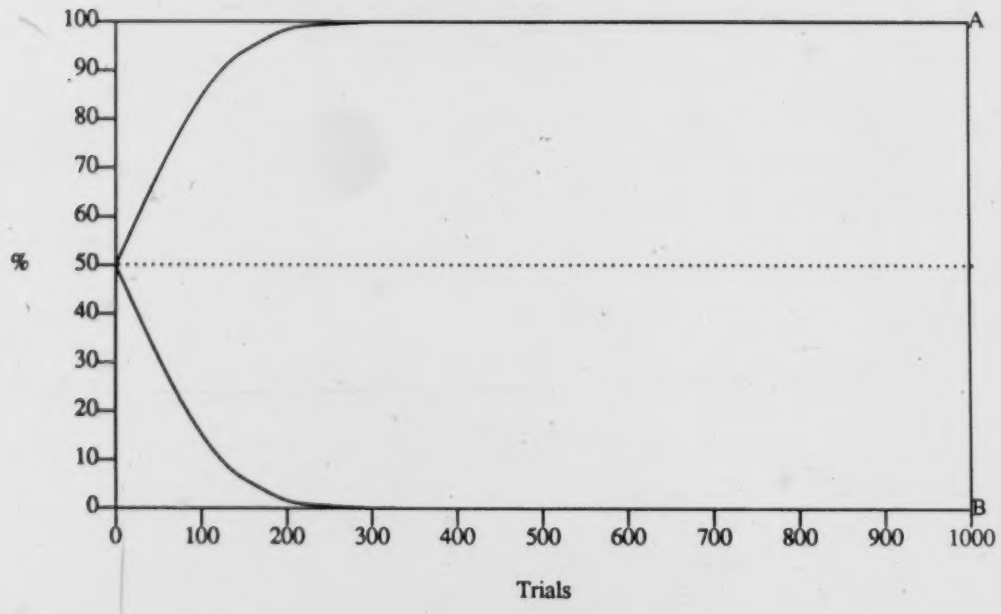

Figure 5.2 2ABP Experiment with Likelihood Selection

5.3.5. Generalization: The K-Armed Bandit Problem

Clearly, a solution to the 2ABP is really of limited usefulness unless it can be extended to consider the more general case of an arbitrary number of slot machines. This has been referred to as the K-Armed Bandit Problem (KABP). It turns out that the algorithm developed previously can be easily extended to the general case of K armed bandits while still retaining asymptotic optimality and fast convergence.

The Generalized Algorithm

The stochastic selection algorithm can be simply extended by giving a weight w_i to $BANDIT_i$ under the following constraints:

$$\sum_{i=1}^{K} w_i = 1, \text{ and}$$

$$0 \le w_1, w_2, \ldots, w_K \le 1.$$

If θ is a random sample from a uniform [0,1] distribution, then a bandit can be selected at each stage by the following rule:

- If $0 \le \theta \le w_1$ then $BANDIT_1$ is selected.
- If $\sum_{j=1}^{i-1} w_j < \theta \le \sum_{j=1}^{i} w_j$ then $BANDIT_i$ is selected, for $i = 2$ to $K-1$.
- If $\sum_{j=1}^{K-1} w_j < \theta \le 1$ then $BANDIT_K$ is selected.

So, the weight $w_i(t)$ of $BANDIT_i$ still gives the effective probability that $BANDIT_i$ will be chosen on any particular trial t.

The weights for each of the K bandits are assigned by a function KPP_i. This function is intended to express, in some sense, the likelihood that $BANDIT_i$ is the best of the K bandits, given the observations. Consider the function

$$kpp_i(t) = \min_{j \neq i} \{PP_{ij}(t)\}.$$

By using this function to assign (unnormalized) weights to the K bandits it can be shown that the resulting

algorithm for the KABP will be asymptotically optimal. An empirical study of its behavior also shows that the algorithm converges rapidly, much like the 2ABP algorithm.

So, at each stage t of the KABP procedure, each bandit is assigned a weight as follows

$$w_i(t) = KPP_i(t) = \frac{kpp_i(t)}{\sum_{j=1}^{K} kpp_j(t)}.$$

Asymptotic Optimality

The asymptotic optimality of the generalized algorithm can be simply shown:

(i) It has been shown for finite t and for arbitrary bandits $BANDIT_i$ and $BANDIT_j$, that $0 < PP_{ij}(t) < 1$. Thus, for finite t, $0 < KPP_i(t) < 1$, since KPP_i simply takes the value of PP_{ij} for some $j \neq i$. So, for $1 \leq i \leq K$, $\lim_{t \to \infty} n_i(t) \to \infty$, meaning that each bandit will receive an unbounded number of trials.

(ii) The property that $\sum_{i=1}^{K} w_i = 1$ arises trivially out of the fact that the weights are normalized.

(iii) Assume, without loss of generality, that $p_2, p_3, \ldots, p_K < p_1$.

As we have seen, the MLE $\hat{p}_i$ is a consistent estimator for p_i, meaning that $\lim_{t \to \infty} \hat{p}_i = p_i$, which implies $\lim_{t \to \infty} \hat{p}_2, \hat{p}_3, \ldots, \hat{p}_K < \hat{p}_1$. So, $\lim_{t \to \infty} kpp'_i(t) = PP_{i1}(t)$, for $i \neq 1$.

It has been demonstrated that, under our assumptions, $\lim_{t \to \infty} PP_{i1}(t) = 0$, and $\lim_{t \to \infty} PP_{1i}(t) = 1$. Thus, yielding the result that $\lim_{t \to \infty} KPP_1(t) = 1$, and for $2 \leq i \leq K$ $\lim_{t \to \infty} KPP_i(t) = 0$.

Convergence

The behavior of this algorithm can be intuitively described as follows: Each of the bandits is given a weight that is proportional to the "degree of certainty" we have that their true proportion of success is greater than that of the apparently best competitor. (The best bandit is compared to the second best bandit.) If some bandit's weight has greatly diminished, then there must be some other bandit who has demonstrated a higher observed proportion of success. Similarly, if a bandit begins to dominate, there must be no other

bandit with a higher observed proportion of success. As more information is acquired from the observations, the greater is our certainty that the observed best bandit really is the best and, hence, the greater its proportion of trials.

Experiment

Compare the graphs shown in Figure 5.3 and Figure 5.4. Both graphs show the results of an experiment with four one-armed bandits: bandit *A* with a payoff proportion of 4/5, bandit *B* with a payoff proportion of 3/5, bandit *C* with a payoff proportion of 2/5, and bandit *D* with a payoff proportion of 1/5. (details of the procedures followed in the KABP experiments can be found in Appendix A1).

Figure 5.3 shows the resulting proportions of trials allocated by the economic selection technique and Figure 5.4 shows the resulting proportions allocated by the likelihood-based selection technique. It is clear upon examining these graphs that the likelihood-based technique converges to the correct proportion — 100% *A*, 0% *B*, 0% *C*, and 0% *D* — at a much faster rate than the economic technique. In fact, the advantage gained by the statistical technique has become even more pronounced as the number of bandits grows.

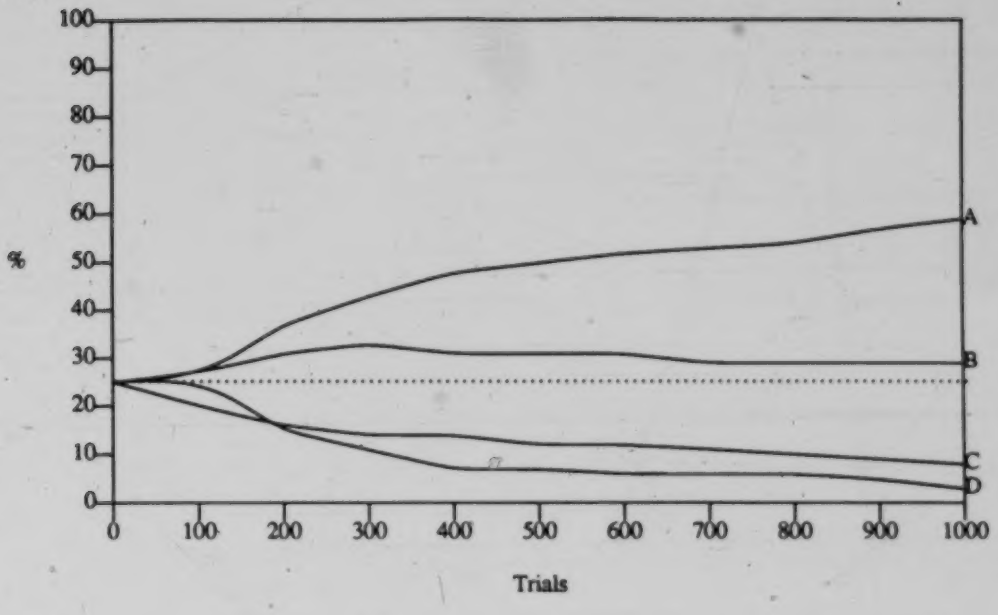

Figure 5.3 KABP Experiment with Economic Selection

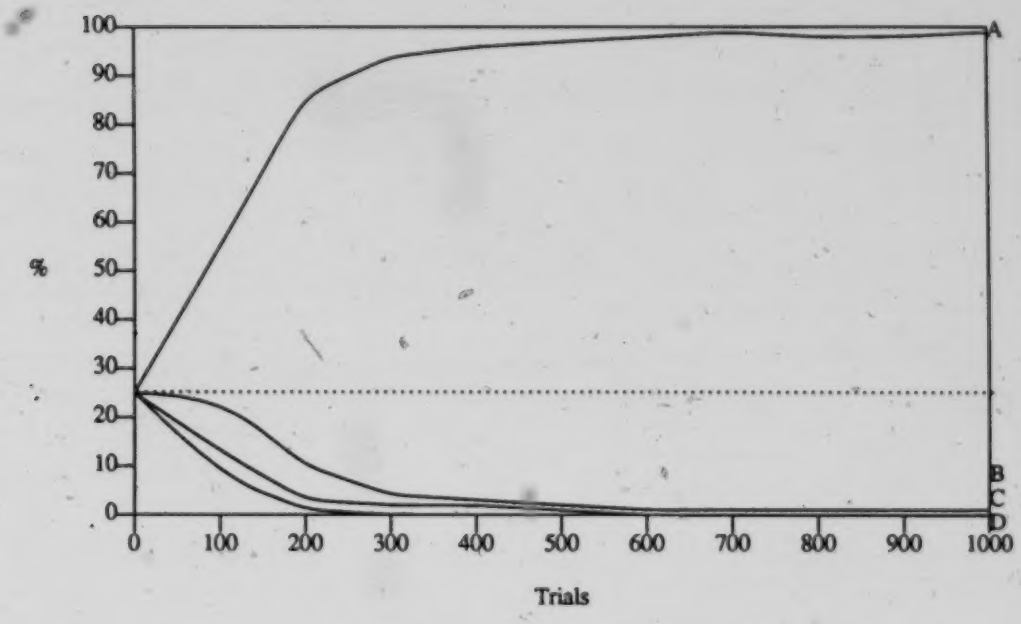

Figure 5.4 KABP Experiment with Likelihood Selection

5.4. Evaluating Fitness

In the previous section an efficient selection algorithm was developed from a detailed reconsideration of the 2ABP. The feedback structure underlying this construction is that of a *Bernoulli trial*. The structure of the feedback must be such that the system receives a response from the environment that indicates whether the results of the preceding action(s) were either "good" or "bad" from the system's standpoint. In the likelihood-based approach, the fitness of a particular classifier is a measure of the likelihood that it is better than its competitors. This is a straightforward property of the classifier's observed proportion of success and the strengths of its competition. The stochastic selection algorithm has a number of desirable properties beyond its sheer speed of convergence. In fact, many of the difficulties encountered by selection algorithms based upon an economic model of fitness are overcome by the likelihood approach taken here.

Proliferation of Parameters

Within the framework developed here, there are no parameters associated with the process of evaluating a classifier's fitness from the feedback provided by the environment. Recall that in the economic evaluation scheme, there were a number of parameters associated with this process: bid ratio, taxation rates and initial strength. It was also argued that this proliferation of parameters constitutes a hindrance to our task of constructing effective learning mechanisms rather than providing a flexibility that can be exploited in a useful way. It is interesting to note that the parameterless selection technique clearly outperforms the heavily parameterized method. A construction that reduces the number of arbitrarily settable parameters without sacrificing the level of learning performance must be seen as advantageous from the point of view of constructing real, working systems.

However, other parameters still remain in the reformulated genetic search technique. For instance, the fact that a system designer must choose a fixed population size and determine the genetic search rates beforehand has not changed. The question remains as to how these parameters in particular may be dynamically managed in a principled, domain independent, and effective manner.

Ad hoc Payoff and Fitness Measures

Again the problem of choosing arbitrary values is avoided by formulating feedback in strictly qualitative terms. Once the designer has sufficiently conceptualized the task domain and arrived at an adequate definition of what constitutes successful and unsuccessful behavior in the domain, no more work is required in terms of quantifying degrees of success or failure. It has been mentioned often that by removing these situations where the system designer is forced to conjure up appropriate values beforehand we greatly improve the chance of the resulting learning system being able to affectively adapt to the environment.

The Scaling Problem

Another fortuitous result of developing a likelihood based selection method is that the scaling problem is overcome simply as a byproduct of the normal behavior of the algorithm. That is, the algorithm tests for absolute differences in the observed success ratios and, thus, is insensitive to the fact that the ratios may be similar on a relative scale. So if the classifier population becomes dominated by legitimately strong classifiers, selection will still favor those classifiers who are stronger than their competitors even if their success ratios are relatively the same. The obvious advantage held by the probabilistic approach over the economic one, in this case, has been clearly demonstrated empirically as well.

Experiment

Consider the graphs shown in Figure 5.5 and Figure 5.6. Both graphs show the results of three experiments with two one-armed bandits.

- The experiment shown by the solid lines utilized one bandit A with a payoff proportion of 3/20, and another bandit B with a payoff proportion of 1/20.
- The experiment shown by the dashed lines utilized one bandit A' with a payoff proportion of 6/20, and another bandit B' with a payoff proportion of 4/20.
- And, finally, the experiment shown by the dotted lines utilized one bandit A'' with a payoff propor-

tion of 19/20, and another bandit *B* '' with a payoff proportion of 17/20.

Figure 5.5 shows the resulting proportion of trials allocated by the economic selection technique and Figure 5.6 shows the resulting proportions allocated by the likelihood-based selection technique. Notice that in each of the three experiments that the payoff proportions of the competing bandits differ by exactly the same amount on an absolute scale. The key difference between the experiments is the fact that the bandits' payoff proportions become closer on a relative scale. Figure 5.5 clearly demonstrates the difficulty that the economic selection technique experiences with proportions that are similar on a relative scale. As the proportions of the competing bandits become closer in relative size, the performance of the economic technique degrades drastically. No similar such degradation in performance is experienced by the likelihood-based selection technique, due to its sensitivity to differences on an absolute scale (Figure 5.6).

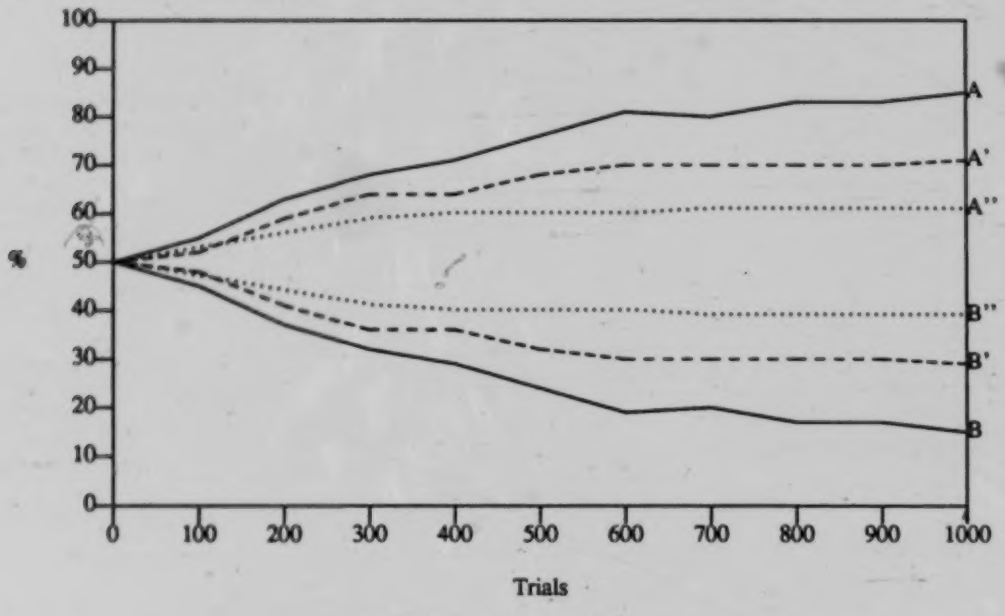

Figure 5.5 2ABP Scaling Experiments with Economic Selection

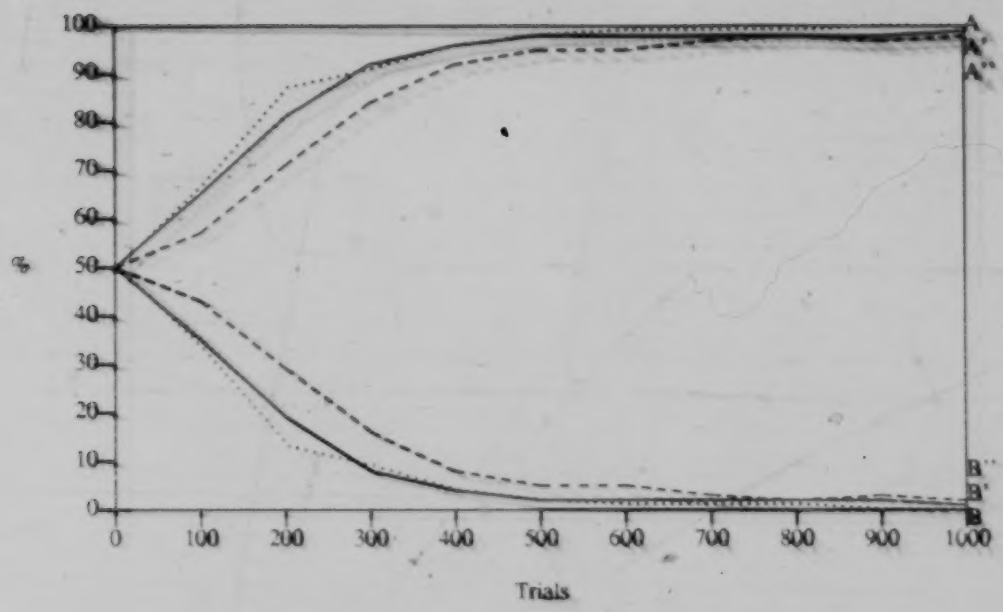

Figure 5.6 2ABP Scaling Experiments with Likelihood Selection

Strength-Time Dependence

In the economic based approach, a classifier gains strength by directly or indirectly achieving sufficient levels of payoff from the environment to offset any loses incurred by participating in the economy. A good classifier will receive a higher rate of payoff than it losses and, over time, its strength will grow to the extent that it can begin to win many competitions. However, regardless of the inherent value that a particular classifier holds for the system as a whole, in an economic based system it will necessarily take time for the value of the classifier to be reflected in a high strength value. In a previous discussion it was mentioned that optimal classifiers may forever be dominated by lesser competitors if only the highest strength classifier wins any bidding competition. Experimental results show that even with a stochastic selection procedure it may take the better classifier an unacceptably long time to overtake a well entrenched, less valuable foe. The advantage held in this respect by the statistical selection technique over the economic technique can be clearly demonstrated with the following experiment.

Experiment

Consider the graphs shown in Figure 5.7 and Figure 5.8. Both graphs show the results of an experiment with two one-armed bandits: bandit *A* with a payoff proportion of 3/4, and bandit *B* with a payoff proportion of 1/4. However, bandit *B* is initialized by giving it the results of having been given 100 trials. That is, 25 successes and 100 trials have been given to *B* before *A* is allowed to compete. So, when the competition begins, bandit *A* — the superior bandit — competes against an inferior, but well entrenched bandit *B*. Figure 5.7 shows the proportion of trials allocated to the bandits by the economic selection technique and Figure 5.8 shows the proportions allocated by the likelihood-based technique.

The economic technique takes time to overcome the skewed initial situation and so the superior bandit is not allocated an appropriate proportion of the trials until a large number of trails have been attempted. Again, the statistical approach appears to be the superior technique in that its performance is not degraded by this skewed initialization. In fact, its performance is improved over the basic experiment simply because of the fact that it has been provided with more information which, in turn, means that it is more certain in allocating trials. So this technique can immediately decide whether a newly introduced classifier actually demonstrates better performance than a well entrenched classifier and immediately allocate trials appropriately.

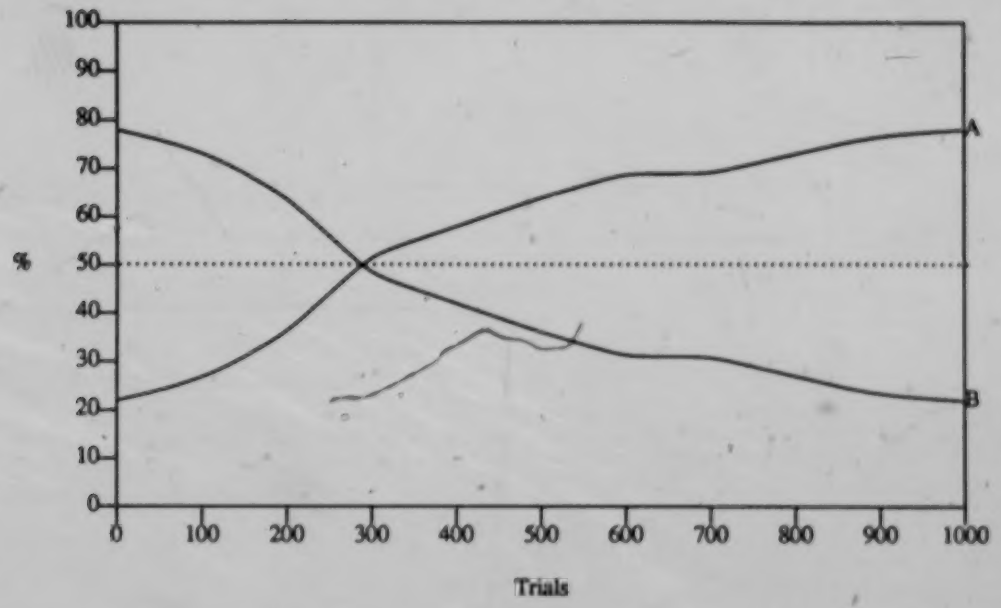

Figure 5.7 2ABP Time Dependence Experiment with Economic Selection

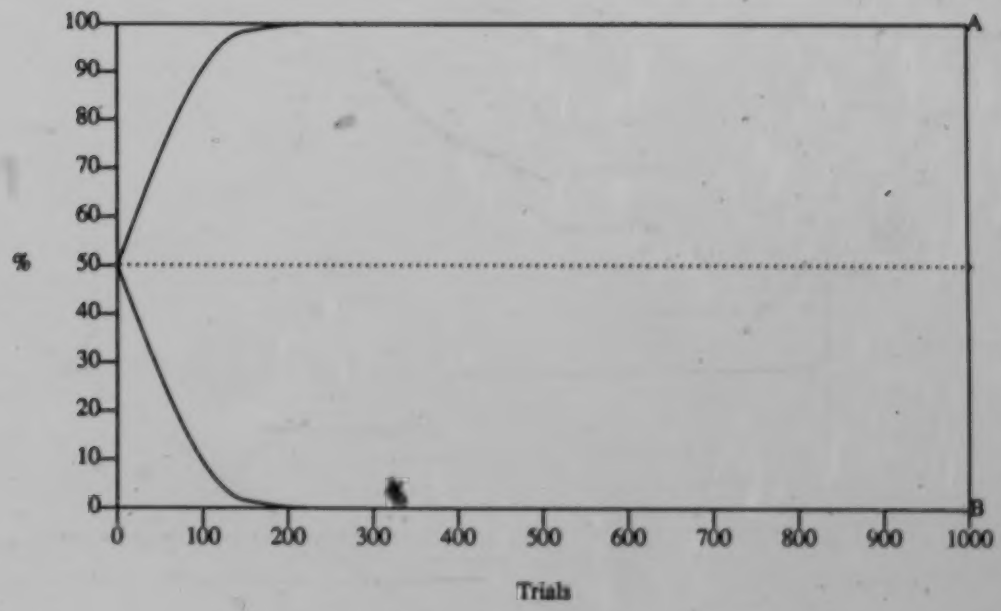

Figure 5.8 2ABP Time Dependence Experiment with Likelihood Selection

5.5. Remarks

In a genetic learning system we are constantly faced with the competing desires of using the knowledge that we already have to our advantage or acquiring more information to increase our knowledge about what is or is not useful. In fact, the 2ABP is simply an abstract formulation of precisely this tradeoff and, consequently, any good algorithm for this problem directly yields a good policy for selecting classifiers to execute. This, of course, was the strategy taken in this chapter. The probabilistic selection technique that was developed allows us to compare the performance of a mature, well entrenched classifier with that of a newly created, apparently successful classifier in a principled, well reasoned way. This method of comparison provides swift and ongoing judgements as to which of a set of competing classifiers is really the best. A fortunate result is that the reformulated algorithm is sufficiently practical to be applied to the classifier selection problem in an obvious and unencumbered way.

Chapter 6

Implementation and Experimental Results

This chapter examines a number of details regarding the actual implementation of genetic learning systems that are intended to overcome the various weaknesses of the traditional classifier system and genetic algorithm technologies by:

- Avoiding some of the representational difficulties of classifier systems (as outlined in chapter 3), specifically by introducing a "disjunctively complete" representation to avoid unnecessary computational explosions.
- Incorporating mechanisms that attempt to overcome or limit some of the difficulties with the genetic search technique (as discussed in chapter 4).
- Incorporating the likelihood-based selection technique that was developed in chapter 5.

A number of versions of a genetic learning system are implemented and tested on a learning problem. First, a credit assignment procedure is developed for use with the likelihood-based selection technique that is based upon the Bernoulli trial formalism. Next, each of the selection algorithms utilized by a genetic learning system are described (conflict resolution, reproduction, and extinction). Finally, a learning task and accompanying representation are set up, followed by a series of experiments which show the performance gains realized through the introduction of some of the techniques developed in this thesis.

6.1. Credit Assignment

Recall that the credit assignment problem is encountered whenever there occurs a situation in which a number of classifiers are executed between feedback episodes. The economic approach offers the bucket brigade algorithm as a method for dealing with the difficulty of assigning appropriate credit to classifiers involved in initiating behaviors that have resulted in environmental payoff. Unfortunately, adapting the bucket brigade mechanism to the likelihood based approach does not appear to be feasible nor desirable. Actually, there appears to be a more natural way to deal with the credit assignment problem within the

scope of the Bernoulli trial formalism.

6.1.1. Bernoulli Trial Sets

The strategy investigated here has been to simply keep a list of the classifiers that are executed between feedback episodes. This list will be referred to as a *Bernoulli trial set* (BTS). When a feedback episode takes place, all of the classifiers that have executed since the last feedback occurrence (those in the BTS) are affected equally.

- If the environment indicates that a success has occurred then each classifier in the BTS has its success counter k_i incremented along with its trial counter n_i.
- If a failure is indicated then each classifier in the BTS has only its trial counter n_i incremented.

Thus, all of the executed classifiers participate equally in what amounts to a single Bernoulli trial for each. The degree to which a particular classifier contributes to the overall success or failure of the system is differentiated from the contributions of other classifiers by the frequency with which its execution is associated with a positive or negative outcome. That is, even though during one particular feedback episode all classifiers are affected equally, their effective contribution to the final outcome is reflected in the frequency of their involvement over a number of such episodes.

For example, consider a classifier system that plays `tic-tac-toe`. If a sequence of moves results in a victory for the system, all of the classifiers responsible for those moves are rewarded equally. Now, over a number of games the classifier that completes the game by placing the third `X` in a row will continually be rewarded whenever it is executed. However, the classifier that is responsible for the first move of a game may often times be involved in losing matches, thus its success ratio will necessarily be lower than the success ratio of the final, winning classifier. This reflects the fact that the final, winning move is, in some sense, more responsible for the final outcome of the game than the initial move is.

The advantage that the BTS approach has over the bucket brigade algorithm is that there is no propagation delay between the time when the first feedback event is encountered to when the first classifiers in

the chain feel even nominal effects of this feedback. By attributing the responsibility of the outcome immediately to all classifiers that were involved there is no such delay present. All classifiers in the chain immediately begin to strengthen if the outcome is desirable, and they are all immediately weakened if the outcome is bad for the system. Yet, even though all classifiers involved are immediately affected by the feedback episodes they encounter, it can still be seen that those classifiers that are more responsible for the outcome ultimately feel a greater effect than those classifiers that are not so frequently associated with the outcome.

So this construction seem to allow for a more efficient evaluation of the chained execution of classifiers that still attributes credit and blame appropriately. Another advantage of using BTSs is that their operation is much simpler to implement, understand, and analyze than the bucket brigade.

6.1.2. Parallel Testing

Another enhancement that has been implemented is parallel testing. This mechanism, again, improves the overall rate at which effective classifier evaluations can be achieved throughout the population. The mechanism is based on a few simple observations about the behavior of a classifier system. Recall that during each execution cycle, stimuli are provided by the external environment in the form of messages. The classifier system then compares all of the classifiers from its classifier population and keeps all possible instantiated matches in a conflict set. Next, a classifier is selected from this set and its action executed, perhaps resulting in some external behavior that affects the state of the environment. The main point is that there might have been a number of other classifiers in the conflict set proposing that this exact same course of action be taken. It would seem to be clear that two classifiers, both performing the exact same action when presented with some identical situation, should both receive credit or blame depending upon the ultimate result of executing the action.

Incorporating this policy into a bucket brigade scheme does appear to be a possibility. However, this would not be nearly as straight forward as it is for BTS approach. Here we must simply add to the BTS all

other classifiers from the conflict set whose instantiated action is identical to that of the selected classifier. This parallel testing policy gathers potentially many more observations with which the true probability of success of a classifier can be more accurately assessed without requiring any extra trials and a trivial amount of extra computation.

6.1.3. Limitations

The main limitation of using the Bernoulli trial structure as a feedback model is that it assumes that a strictly qualitative judgement of the classifier system's behavior is available. Of course, this may not always be the case. There are certainly problems that are quantitative in nature so that a qualitative assessment is not feasible. One possible extension of the ideas presented here might be to generalize the feedback and selection mechanisms to handle quantitative measures. However, allowing for quantitative measures may reintroduce the problems associated with *ad hoc* values. Anyways, a qualitative approach can still be quite general, and the mechanisms so far described can be generalized to handle a much wider range of interesting domains by allowing for multi-dimensional, yet still qualitative, feedback.

6.1.4. Multidimensional Feedback

Performance in the task domain can often be divided into a few distinct aspects. How this division must occur can be influenced by a number of factors. One of the most important reasons that the measure of task performance might be split up into separate dimensions has to do with the problems of credit assignment. For example, in the `tic-tac-toe` domain, one might view the performance of the system as comprised of the legality/illegality of its moves and whether it wins/loses/draws the games. The important distinction between these two dimensions is the fact that in the case of a legal/illegal move only the single action that caused the condition is responsible, whereas, in the case of a won/lost/drawn game, the entire preceding sequence of actions was responsible as a whole for the final outcome. This is an important distinction from the point of view of appropriately assigning credit or blame to the classifiers in the system.

Given that the measure of task performance can be appropriately divided into a few separate dimensions, there remains the problem of accounting for multiple feedback dimensions in the likelihood-based selection algorithms developed previously. The approach considered here has been to first calculate partial weights for all of the competing classifiers, one dimension at a time. Once each classifier has a separate weight for each dimension the final weight is calculated by simply multiplying each of the partial weights together.

In practice, this procedure produced good results but with some limitations. Search performance was best when there existed classifiers that could exhibit good performance in all aspects of the domain task. I suspect that the search would be much less effective when no one classifier could encode behavior that was successful along all dimensions of the task. In reality, the search technique was not developed with multiple performance dimensions in mind. The method of multiplying the individual likelihoods to arrive at an evaluation of a classifier's overall fitness arose strictly from intuitions and pragmatic concerns but this procedure is not justified in any real, formal sense and must be considered *ad-hoc*.

6.2. Selection Algorithms

There are three distinct selection mechanisms used in constructing a genetic learning system. A selection mechanism is utilized for the conflict resolution process, the classifier reproductive process and classifier extinction.

6.2.1. Conflict Resolution

The conflict resolution phase of the classifier system execution cycle involves a competition among all of the classifier instantiations in the conflict set for the right to execute their respective actions. The algorithm developed here utilizes a weighted stochastic selection procedure like that described in a previous section. Each classifier in the conflict set is given a weight that is proportional to the likelihood that it is the classifier that yields the best chance of executing a successful action. In the case where feedback is provided in a multidimensional form, the weights for each dimension are calculated independently and then the

results multiplied together for each classifier. Once the weights have been assigned to each classifier, one of them is selected by the stochastic procedure according to their weights. This classifier and all other classifiers in the conflict set that specify the same action are saved in a BTS for credit assignment purposes.

This particular selection procedure for conflict resolution has a distinct advantage over traditional conflict resolution selection procedures in that it more effectively balances the competing needs of exploration and exploitation (this advantage was clearly demonstrated in chapter 5). As more information about the success ratios of the various classifiers is acquired, we become more certain that the classifiers observed to have been the best really are so. The proportion of trials allocated to these successful classifiers grows in direct correspondence to this increased certainty. Thus, if a superior classifier is introduced at some point of the search, its observed ratio of success will eventually demonstrate this fact with ever increasing certainty and, hence, it will eventually dominate its inferior competitors to the point where it wins virtually all conflict resolution competitions. In fact, if we assume that no classifier is deleted from the population, then it is clear that the classifier system is guaranteed to achieve its optimum level of performance in the limit. Of course, it is not practical to keep every classifier in the population indefinitely, meaning that at some stage of the search classifiers will have to be deleted from the population. So there is always a non-zero chance that an optimal classifier could be deleted from the population before it could adequately demonstrate its value. This, however, is an unavoidable risk.

6.2.2. Reproduction

The reproductive phase drives the heuristic search through the classifier space and, as such, it provides the impetus behind the genetic search technique. It has been previously mentioned that the heuristic underlying the entire genetic search process can be stated as follows:

> If some classifier has demonstrated good task performance, then a similar classifier may also demonstrate good performance and, perhaps, even better performance.

The definition of similarity is dependent upon the genetic operators that are used to construct progeny from

the parent classifiers.

The reproductive process involves a global competition amongst the entire classifier population for the right to become progenitors for the next generation of offspring. Normally, this selection process is biased, favoring strong classifiers over weak classifiers. Underlying this bias is the assumption that the offspring of superior parents are likely to be fit, more so than the offspring of inferior parents.

The algorithm used to select parents for the next generation of classifiers is again constructed from the likelihood-based weighted stochastic selection technique developed earlier. That is, all of the classifiers in the population compete for the right to reproduce, each classifier receiving a weight that is proportional to the likelihood that it is superior to its competitors. This procedure provides a strong bias towards selecting as parents those classifiers that have proven to be superior than the remainder of the population.

Multidimensional fitness measures are dealt with in much the same manner as with the previous selection processes. Once the appropriate values have been arrived at for each dimension, the final combined weight for a classifier is obtained simply by multiplying the values from each dimension together.

6.2.3. Extinction

Under the fixed population size assumption, whenever a new classifier is to be introduced to the classifier population, some existing classifier must be removed to make way for its insertion. A classifier extinction phase immediately precedes any insertion phase in the execution cycle. This extinction phase normally involves a global competition for survival amongst the entire classifier population. It has been observed that whenever any classifier is removed there is a non-zero chance that it may have been an important one to retain. The real issue is how is the risk of prematurely removing an important classifier from the population to be minimized?

The mechanism to be used here is just a simple, deterministic selection of the classifier that happens to posses the lowest observed success ratio at the onset of the extinction phase. So the apparently fittest classifiers survive each extinction phase. The risk of losing superior classifiers is small because they will

normally outperform the inferior members of the population and, hence, be spared during the extinction phase. However, there is a non-zero likelihood that a superior classifier will, by chance, demonstrate apparently inferior behavior in the short run and be removed as a result. This possibility is unavoidable, in principle, but this deletion policy is a reasonable one in that superior classifiers are not likely to be removed.

This selection technique has been simply extended to deal with multidimensional fitness measures by simply prioritizing the dimensions and progressively filtering out the worst classifiers in each dimension until a set of worst classifiers remains. One of these classifiers in the worst set is then targeted for removal.

6.3. `Tic-Tac-Toe` Legal Move Experiments

Recall the `tic-tac-toe` legal move problem that was presented and discussed in chapter 3. This problem shall be used to demonstrate the performance characteristics of a number of genetic learning systems that employ different mechanisms. These legal move experiments will utilize a representational scheme that avoids the problems associated with the limited capabilities for representing disjunction possessed by traditional classifier representations. Instead of encoding the various domain values as bit patterns, the input messages are simply composed of lists of symbols. For example, the message `(X B O B X O B B B)` would denote the following board configuration:

X	B	O
B	X	O
B	B	B

The classifiers have condition strings with nine locations, one for each square on the `tic-tac-toe` board, to match against board configuration messages. Each of these locations contain some (non-empty) subset of the set of possible marks `{X,O,B}`. A condition string will match a configuration message if for each location i, the subset at position i in the condition string contains the symbol at position i of the configurations message. Such a representation scheme can express any disjunctive combination of the symbols and, hence, any potential explosion (related to limited disjunction) in the number of classifiers needed

to express solutions is avoided. The action string will merely designate the square in which an `X` is to be placed, where the squares are addressed as follows:

0	1	2
3	4	5
6	7	8

Given such a construction, a classifier that encodes the rule *if the center square is blank, then place an X in the center square* would be expressed by:

`(X O B) (X O B) (X O B) (X O B) (B) (X O B) (X O B) (X O B) (X O B)` $\rightarrow$ `4`

With this representation, a complete solution to the legal move problem requires a genetic learning system to discover and maintain nine distinct classifiers in the form of the above classifier, one for each square.

Experiment

Figure 6.1 depicts a graph of the performances on the legal move problem by two different genetic learning systems, a traditional economics-based system (curve E), and a system that incorporated the likelihood-based mechanisms developed here (curve L) (details of the procedure followed in performing `tic-tac-toe` legal move experiments can be found in Appendix A2). The measure of performance is simply the proportion of legal moves to illegal moves made by the systems as they adapt. It can be clearly seen from Figure 6.1 that the likelihood-based system outperforms the economic based system.

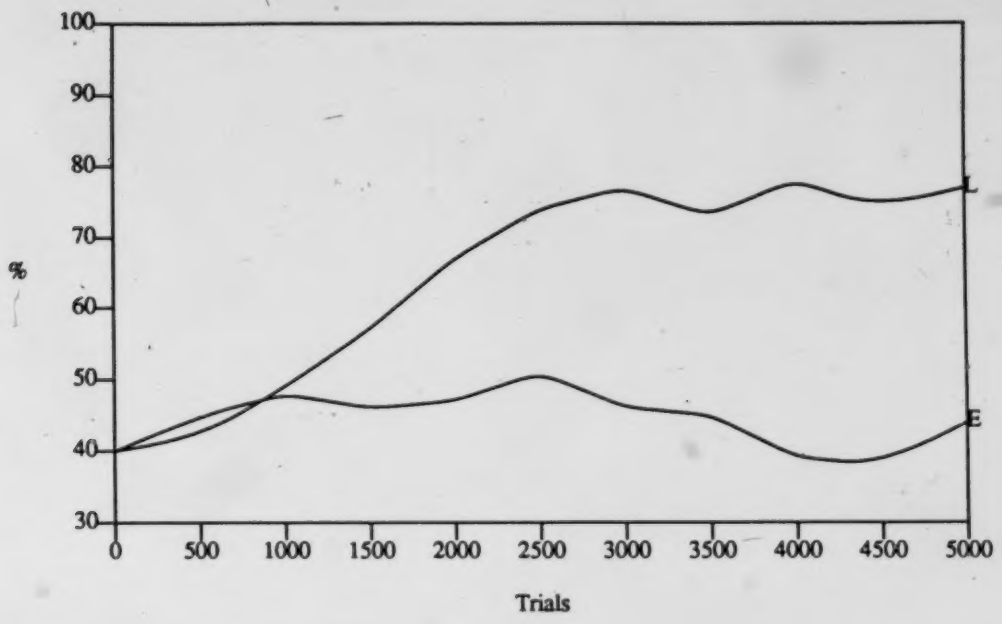

Figure 6.1 Legal Move Experiment - Economic vs. Likelihood

However, notice that in each case, the performance of the system plateaus well before a high level of success is achieved. The reasons for this leveling off of performance can be seen by examining Table 6.1 and Table 6.2. These tables are intended to show, in some sense, how the "quality" of the classifier population changes as the systems adapt. Table 6.1 shows what proportion of a perfect solution (nine generalized classifiers, one for each square) was discovered by the economic-based system at various stages of its search. This table also shows what percentage of each of the sub-solutions, one for each square, was discovered, and (in small type) what proportion of the total population belongs to each of these sub-populations. Table 6.2 shows the corresponding information for the likelihood-based system.

Trials	% Solution	Percentage of Solution for each Sub-Population (Percentage of Population in each Sub-Population)								
		0	1	2	3	4	5	6	7	8
1000	37	28	56	37	72	31	7	57	40	10
		6	15	8	15	9	2	19	12	5
2000	33	0	9	67	96	0	0	89	36	0
		0	3	6	64	1	1	22	2	1
3000	33	0	0	82	100	7	0	89	0	17
		1	0	6	50	1	0	33	0	8
4000	38	0	0	89	100	0	0	99	0	56
		0	1	14	21	0	0	55	0	10
5000	25	10	0	18	100	0	0	100	0	0
		2	1	4	11	1	1	79	0	1

Table 6.1 Legal Move Experiment with Economic Selection

Trials	% Solution	Percentage of Solution for each Sub-Population (Percentage of Population in each Sub-Population)								
		0	1	2	3	4	5	6	7	8
1000	40	39	67	59	27	54	41	18	28	30
		9	8	25	5	13	16	5	7	3
2000	52	51	63	76	2	35	65	56	73	48
		13	6	30	4	7	16	6	12	6
3000	59	78	76	91	16	10	66	31	78	86
		23	5	38	2	2	6	4	12	7
4000	61	96	78	99	34	0	63	7	86	89
		25	2	42	2	0	3	1	14	9
5000	53	98	17	96	35	15	31	0	88	97
		33	3	34	3	0	3	3	13	8

Table 6.2 Legal Move Experiment with Likelihood Selection

These tables demonstrate, again, that the likelihood-based system discovered a higher proportion of the solution, but notice that in both cases, the proportion of the total solution discovered has begun to diminish. The sub-population sizes indicate why this has occurred. Both of the systems have over-converged. That is, they have converged to good solutions in a few of the sub-populations, but systems have completely lost the representatives from other sub-populations. Thus, they have lost diversity that was necessary for attaining and maintaining an optimal solution.

Further experiments of this nature are presented in the next section that attempt to improve on these

systems and overcome some of their weaknesses.

6.4. Extensions

Recall that the convergence difficulties encountered by the previous experiment were presented and discussed in chapter 4. In this section, a number of mechanisms are developed which are an attempt to overcome some of these major weaknesses of the selection algorithms developed here and of genetic algorithms in general. First, an enhancement to the likelihood-based approach is presented and its performance investigated.

6.4.1. Ubiquity

It is important for the genetic learning system to not only discover a solution that exhibits optimal task performance, but also the system must encode its behavioral capacities with a limited, if not minimal, number of classifiers. An obvious way to capture this need is to attempt to discover the few most general classifiers that exhibit the desired performance. Machine learning systems almost universally encode a bias towards generality in their structures, and this is true of genetic algorithms as well.

The bias towards generality is captured in a Holland type classifier system through the scheme that protects generalists in the bidding mechanism. The intent is to prevent general classifiers from potentially losing their strength as quickly as their specialists. So, generalists will tend to acquire greater strengths, thus biasing the conflict resolution and reproductive selection processes in their favor [Hol85, Hol86, Hol87]. What is not clear from Holland's treatment is that there is no real need for the bias in the conflict resolution phase, and that the presence of such a bias is merely an artifact of the economic model being used to estimate classifier fitness. The ultimate prevalence of generalists in a population is the result of the bias in the reproductive phase only.

Notice that there is no bias in favor of generalists in the conflict resolution mechanism developed here. That is, classifiers are selected for execution based solely upon their demonstrated utility to the system. So the issue at hand is how to construct a reproductive mechanism that appropriately biases the search

around successful and general classifiers.

Given the feedback and fitness models that have been developed up to now, the approach towards biasing the search in favor of general classifiers is to introduce another fitness dimension — *ubiquity*. Ubiquity is defined to be the observed ratio of matchings to cycles. The total number of cycles that a classifier has existed and the total number of successful matches during that period are kept for each classifier. That is, instead of using an artificial measure of a classifier's syntactic generality an operationalized measure of a classifier's real usefulness to the system is utilized. Once an ubiquity dimension is defined, the normal mechanisms for combining multidimensional fitness measures will automatically bias the reproductive selection in favor of those classifiers that have demonstrated a high rate of utility.

Experiment

Figure 6.2 shows the results obtained from a legal move experiment that utilized a genetic learning system that incorporated the ubiquity mechanism (curve U) along with the results obtained from the previous experiment that utilized basic likelihood selection (curve L).

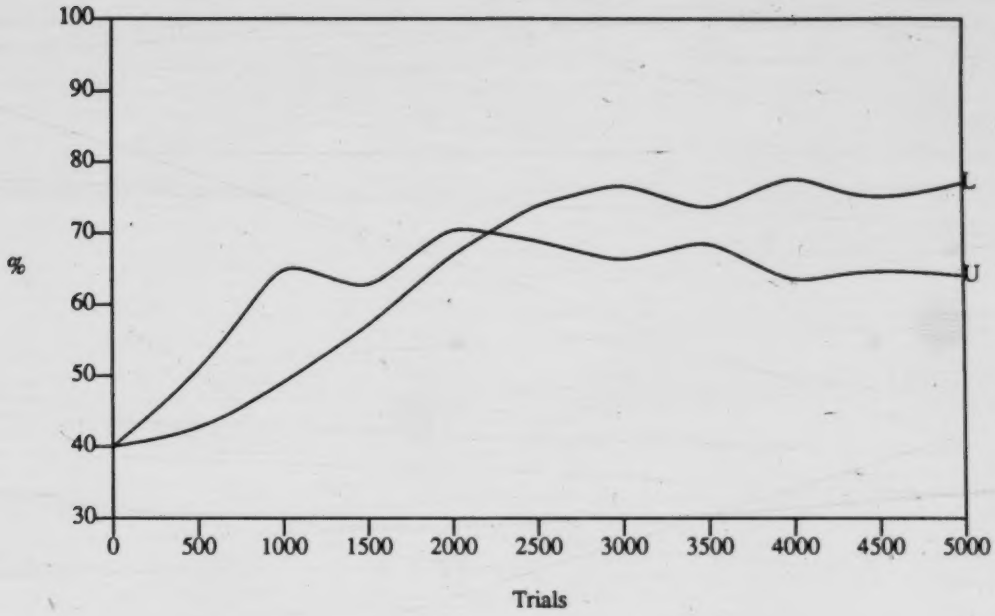

Figure 6.2 Legal Move Experiment - Likelihood vs. Ubiquity

Trials	% Solution	Percentage of Solution for each Sub-Population (Percentage of Population in each Sub-Population)								
		0	1	2	3	4	5	6	7	8
1000	40	51	77	30	87	7	34	0	35	41
		5	17	7	35	4	12	3	5	4
2000	47	37	78	23	100	36	72	22	18	34
		5	6	2	61	7	9	4	3	3
3000	43	57	78	0	100	0	100	12	15	20
		2	3	2	56	1	31	2	1	1
4000	40	27	53	0	100	22	100	0	8	48
		2	2	0	49	1	40	2	1	3
5000	37	21	28	0	100	4	100	0	32	44
		1	1	0	48	3	41	1	1	3

Table 6.3 Legal Move Experiment with Ubiquity Bias

The introduction of the ubiquity mechanism apparently diminished the performance of the learning system a great deal! However, upon examining Table 6.3 and noticing that the ubiquity mechanism actually began to improve rapidly but then plateaued early, we can see that the system actually converges much faster than

the previous two systems, leading to quicker over-convergence. Most of the diversity is lost almost immediately, while the system quickly discovered optimal sub-solutions in two of the nine sub-populations — at the expense of the other sub-populations. This next section attempts to incorporate a mechanism that limits over-convergence.

6.4.2. Dominance

It has been mentioned earlier in chapter 4 that the simple algorithm for classifier extinction leads to some difficulties when the environment presents a number of distinct niches, all of which should be taken advantage of. That particular discussion illustrated how the technique of simply removing the worst classifier from the population during each extinction phase, regardless of its form, tends to rob the population of its diversity, thus, leading the classifier system into a convergent state where only a single environmental niche is exploited.

The approach taken here, in an attempt to cope with convergence difficulties, is in the same spirit as the approach taken to favor generalists — through the introduction of another fitness dimension that will be referred to as *dominance*. Dominance is defined to be the observed ratio of selections to matchings. The total number of times that a classifier has been selected in the conflict resolution cycle and the total number of successful matchings are kept for each classifier. Once the dominance dimension is defined, the normal mechanism for combining multidimensional fitness measures automatically biases the selection mechanisms to favor those classifiers that win a large proportion of their competitions.

The operation of this mechanism in combating convergence is twofold.

- Dominance limits over-convergence in the presence of a predominantly successful sub-population. When some particular form of classifier is discovered to perform well in the environment it acquires a high fitness rating, thus, increasing its reproductive advantage. When this occurs, a number of classifiers similar in form will be introduced to the population through the reproductive process. If the general form of these classifiers is particularly successful a sub-population will grow around this

form. Members of the same sub-population (i.e. classifiers with similar forms) will tend to match the same set of environmental stimuli and so they will compete with each other on successive conflict resolution cycles. As the sub-population grows, the chances of any one classifier winning a competition diminishes and, hence, the members of the sub-population receive a low dominance rating.

- Dominance limits the loss of classifiers that are successful in specialized niches. A classifier that is reasonably successful at exploiting some particular niche but does not belong to a large, well entrenched sub-population, will not have many competitors in those situations where it is matched. In which case, it will win most of its competitions and, hence, receive a high dominance rating.

Experiment

Figure 6.3 shows the results of a legal move experiment with a genetic learning system that incorporates both the ubiquity and dominance mechanisms (curve D) along with the previous results from the system with ubiquity alone (curve U).

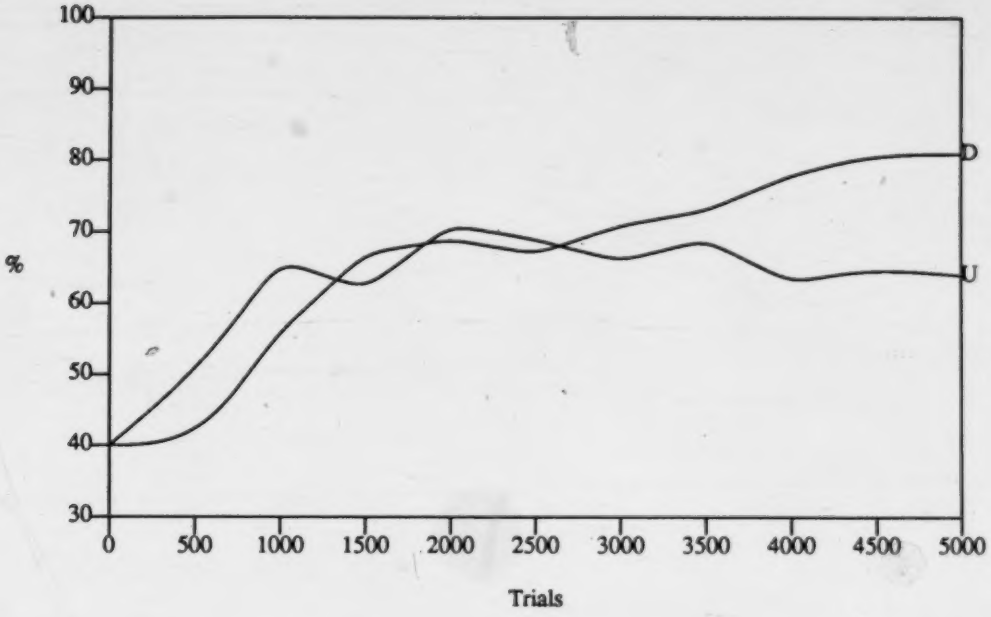

Figure 6.3 Legal Move Experiment - Ubiquity vs. Dominance

Trials	% Solution	Percentage of Solution for each Sub-Population (Percentage of Population in each Sub-Population)								
		0	1	2	3	4	5	6	7	8
1000	43	40 7	33 4	44 15	36 8	40 10	56 16	51 9	18 6	67 15
2000	58	67 11	44 4	67 17	33 8	33 9	67 11	79 11	39 7	97 23
3000	68	67 7	54 8	71 11	64 11	48 12	67 10	93 21	67 6	82 15
4000	73	60 8	80 11	74 13	67 10	52 15	66 15	91 14	67 4	96 9
5000	69	46 5	100 7	71 10	57 11	48 24	58 21	92 8	50 9	100 6

Table 6.4 Legal Move Experiment with Dominance Bias

The performance of the system with dominance is superior to that exhibited by the system incorporating only an ubiquity mechanism. The introduction of the dominance mechanism appears to have reduced the plateau effect experienced by the previous systems. Table 6.4 shows that the dominance mechanism does,

in fact, limit over-convergence so that the predominantly successful sub-populations do not grow at the expense of the other sub-populations. Upon examining Table 6.4, it seems that a sub-population grows in size while its membership continues to improve, but the sub-population will begin to shrink if progress does not continue to be made. In this respect, the dominance mechanism is quite a powerful technique for improving the performance of the genetic search in a domain-independent manner. However, a disturbing property of Table 6.4 is that, periodically, the system appears to lose important and useful classifiers. This problem would seem to stem from the *ad hoc* way in which the multiple fitness dimensions are combined in the extinction phase of the genetic search.

6.4.3. Sub-Population Separation

A difficulty that presents itself when we simply allow the entire classifier population to compete equally for the right to reproduce is that interference problems can arise. As was discussed, the problem is that classifiers that exploit different niches in the environment normally possess forms that are significantly assimilar. When two such classifiers are recombined, the resulting offspring will likely not be useful in either niche because it contains building blocks from what might be two necessarily distinct sub-solutions.

Interference difficulties can be avoided to some degree by introducing a simple sub-population separation scheme to the reproductive process. The simple, static policy implemented here ensures that two classifiers possessing distinct action strings are never the joint progenitors for an offspring classifier. So the classifier population is statically divided into separate species, one for each possible action, and mating is restricted to occur only between members of the same species.

The disadvantage of this approach can be seen if we consider that two distinct actions may be useful in much the same type of situation. If two different actions are effective in the same situations then some of the search effort will be duplicated between these two sub-populations. The argument is that, generally, distinct actions are effective in a distinct set of situations, otherwise distinguishing between the actions becomes unimportant.

This sub-population separation scheme actually demonstrates a performance improvement in empirical tests. A major source of interference between species that exploit completely different niches is avoided. However, this mechanism constitutes only a partial solution to the interference problem, far from the general goal of achieving dynamic and effective speciation.

Experiment

Figure 6.4 shows the performance obtained by a system that utilized dominance and ubiquity mechanisms, with nine, fixed size sub-populations, one for each square (curve *S*). These results are compared with the results from the previous experiment that utilized only dominance and ubiquity (curve *D*).

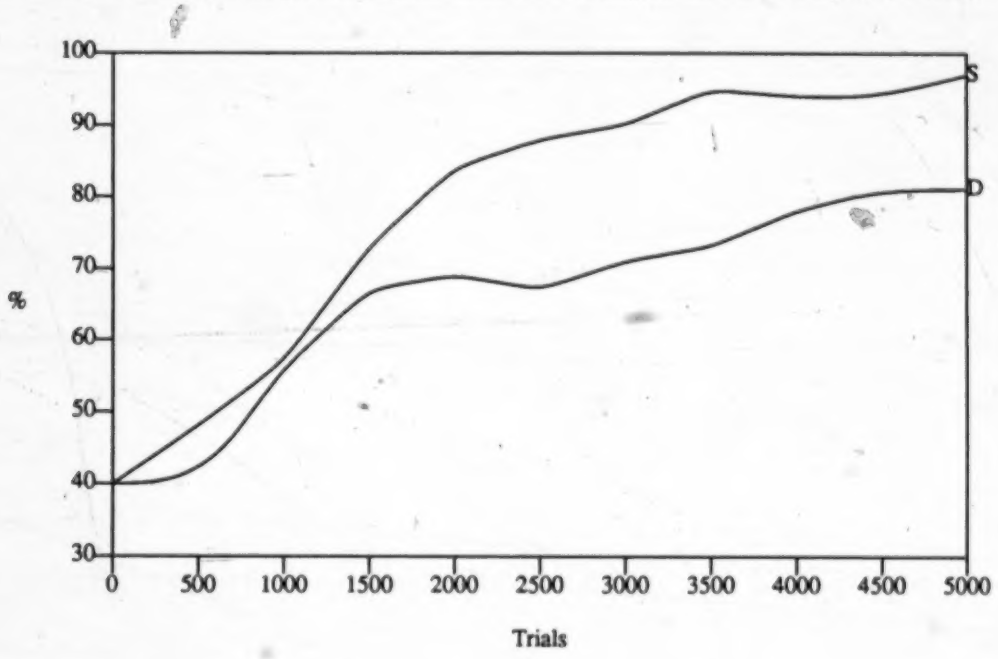

Figure 6.4 Legal Move Experiment - Dominance vs. Sub-Populations

Trials	% Solution	Percentage of Solution for each Sub-Population								
		0	1	2	3	4	5	6	7	8
1000	44	43	7	51	33	56	76	66	16	48
2000	77	68	69	84	43	67	98	87	89	87
3000	85	78	82	89	59	73	100	100	100	89
4000	93	78	99	97	79	91	100	100	100	95
5000	98	97	100	93	97	91	100	100	100	100

Table 6.5 Legal Move Experiment with Sub-Populations

A pronounced performance improvement is realized. This is due to the fact that there is no longer any interference between the separate searches that are attempting to discover distinct sub-solutions independently from one another. This vast improvement in the effectiveness of the genetic search is corroborated by Table 6.5.

Chapter 7

Concluding Remarks

The general objective of this research was to construct an effective, domain-independent learning system. Genetic learning systems were considered as an appropriate approach to take towards constructing such black box learning systems. A more specific goal has been to examine the techniques underlying this genetics inspired approach with the intent of ultimately improving upon the capabilities of such systems. To that end, three main issues were pursued throughout the course of this thesis.

- The representional adequacy of classifier systems.
- The difficulties experienced by genetic algorithms when attempting to discover multiple classifier solutions.
- The tradeoff between exploiting the current knowledge of a system to achieve the best possible performance in the short run and exploring new behaviors in an attempt to improve upon this performance in the long run.

The first issue to be investigated was the representational adequacy of classifier systems. A number of small difficulties associated with using classifier systems were immediately observed. However, upon further examination a number of properties of the representation were discovered to have a profound effect upon the way in which solutions to problems could be represented. The most serious of these was the limited expressiveness of the representation. Not all of the possible disjunctive combinations of values could be represented in a single field, creating the potential for a combinatorial explosion in the number of classifiers needed to represent a solution. Such an explosion in the size of a solution would have an equally drastic effect upon the efficiency of a genetic search that attempts to discover this solution.

Another major topic of investigation pursued throughout this thesis was an examination of the difficulties encountered by the genetic learning technique when it attempts to discover multiple classifier solutions. This was pursued both through a survey and discussion of the processes underlying the prob-

lems, and through the construction and empirical testing of mechanisms that attempt to remedy these difficulties. Two of the major problems confronting the genetic search technique were convergence and interference. Although these two difficulties pose a serious obstacle to obtaining an effective genetic search, techniques have been developed to combat them with some (albeit limited) success. Another difficulty, independence, is more serious in the sense that its effects cannot be overcome even in principle. The main point is that the more "targets" the genetic search must discover, the more effort is required to discover all of them, notwithstanding suitable techniques that overcome convergence and interference difficulties. In this respect, the fact that the classifier system may require a larger than necessary number of classifiers to represent solutions appears to condemn the representation as being unsuitable for use with a genetic search.

A final, important topic of investigation was the detailed reconsideration of the tradeoff between exploration and exploitation in its abstract form — the 2-Armed Bandit Problem. A careful examination of this problem led to the development of an improved, statistically motivated selection technique that demonstrates a substantial performance advantage over the traditional economic technique in a series of empirical tests. As a selection technique for genetic learning systems, the likelihood approach also avoids most of the weaknesses of the traditional technique: proliferation of parameters, *ad hoc* payoffs, scaling, and strength-time dependence.

Results

The main result of this research has been the construction of a genetic learning system that incorporates all of the suggestions and procedures that were developed throughout. This implementation is built upon a number of results.

- The development of a statistically motivated stochastic selection technique that effectively balances the tradeoff between exploration and exploitation. This, in turn, led to the development of an accompanying credit assignment scheme that is superior to the traditional, economics-based technique.

- The development of mechanisms that demonstratably overcome some of the traditional weaknesses of the genetic search technique. The ubiquity mechanism to help the genetic search better discover general, more concise solutions. The dominance mechanism to combat the tendency for the genetic search to over-converge. And, finally, sub-populations to eliminate interference difficulties.
- The development of a simple, classifier-like representation scheme that overcomes some of the difficulties of traditional classifier representations. Most importantly, this new representation scheme is "disjunctively complete". A nice feature is that it tends to provide better human factors, as well.

Under empirical test, the genetic learning system that incorporated the suggestions and procedures developed in this thesis demonstrates significant improvement in performance over the traditional form of genetic learning system. This improvement does constitute progress towards the general goal of constructing effective black box machine learning systems.

Future Directions

In the previous discussions, a number of issues have been raised that have yet to receive an adequate treatment. These issues raise a number of difficult problems that remain to be solved in a completely satisfying manner. Although some progress was made towards the general goal, there still remain a large number of weaknesses with the genetic learning system that was developed.

Multiple fitness dimensions have been seen to be an important generalization of the Bernoulli trial formalism for credit assignment. Unfortunately, the likelihood-based selection technique was not developed with this generalization in mind and, consequently, several *ad hoc* procedures were utilized when having to combine multiple fitness measures. This led to a number of difficulties when extra fitness dimensions were introduced as extensions to the basic genetic search mechanism. For instance, with the ubiquity bias, there was a tendency for the genetic algorithm to over-generalize its classifiers. The ubiquity dimension dominated the search process mainly due to the fact that more information could be acquired in this dimension than any other dimension and, hence, the certainty levels in this dimension grew at a much

faster rate. Another difficulty that appeared with the introduction of the dominance dimension was the tendency for the genetic learning system to periodically lose valuable classifiers. The mechanisms utilized to combine these various fitness dimensions did not adequately balance the tradeoff amongst these competing factors.

Of all the factors that can influence the performance of a genetic algorithm that is searching for a multiple classifier solution, the most important, based upon the experimental results, seems to be interference. Recall the tremendous performance improvement that was obtained by separating the classifier population into distinct sub-populations, one for each sub-solution. This is because *global* optimization is a strength of the genetic search technique [Bri80] and because interbreeding between different "species" of classifiers tends to impede, not enhance, the search. Of course, the form of static separation of the classifier population that was used to achieve the improved performance violated the assumptions about black box learning systems. Dynamically developing appropriate species in a population in an efficient and effective manner is a difficult, open problem in genetic learning system research.

An important point that was made throughout was the fact that the proliferation of parameters normally associated with the genetic search technique is a hindrance when applying genetic learning systems to application domains. The fact that the likelihood-based selection technique eliminated a number of parameters was seen as a strength of the approach. However, a few parameters remain. The application developer must still provide the system with an appropriate population size, crossover rate, and mutation rate. Again, developing a mechanism that dynamically adapts the population size and search rates appropriately and effectively remains another difficult open problem.

A basic assumption underlying all of the developments in chapter 5 was that the inherent probability of success possessed by a one-armed bandit (or a classifier for that matter) was fixed. That is, the probability of obtaining a successful payoff with a given bandit or classifier did not change. This could be seen as a restrictive assumption in that many interesting machine learning problems involve a changing environment. For example, the machine could be attempting to learn how to successfully play some game against an

opponent *who is adapting as well*. Thus, certain behaviors that were successful at one point in time, may not continue to be all that successful in the future. The selection mechanisms developed in this thesis are not well equipped to cope with such an environment. This would certainly be a useful extension to consider.

I would hope that, as a result of this research, it can be seen that black box learning techniques, in general, and genetic learning systems, in particular, are an important area of machine learning research wherein a number of fundamental issues remain to be addressed.

References

[Bak85] BAKER, J.E., Adaptive selection methods for genetic algorithms, *International Conference on Genetic Algorithms and their Applications*, 1985, 101-111.

[Bat80] BATHER, J., Randomised allocation of treatments in sequential trials, *Advances in Applied Probability 12*, (1980), 174-182.

[Boo85] BOOKER, L.B., Improving the performance of genetic algorithms in classifier systems, *International Conference on Genetic Algorithms and their Applications*, 1985, 80-92.

[BrG87] BRIDGES, C.L. and GOLDBERG, D.E., An analysis of reproduction and crossover in a binary-coded genetic algorithm, *Second International Conference on Genetic Algorithms and their Applications*, 1987, 9-13.

[Bri80] BRINDLE, A., *Genetic algorithms for function optimization*, Doctoral Thesis, University of Alberta, Edmonton, Alberta, 1980.

[De85] DE JONG, K., Genetic algorithms: A 10 year perspective, *International Conference on Genetic Algorithms and their Applications*, 1985, 169-177.

[De87] DE JONG, K., On using genetic algorithms to search program spaces, *Second International Conference on Genetic Algorithms and their Applications*, 1987, 210-216.

[DiM83] DIETTERICH, T.G. and MICHALSKI, R.S., A comparative review of selected methods for learning from examples, in *Machine Learning*, MICHALSKI, R.S., CARBONELL, J.G. and MITCHELL, T.M. (ed.), Morgan Kaufmann, Los Altos, California, 1983, 41-81.

[Fen88] FENSKE, K., An interactive classifier programming language, Masters Thesis, University of Alberta, Edmonton, Alberta, 1988.

[For81] FORGY, C.L., OPS5 user's manual, Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1981.

[GoR87] GOLDBERG, D.E. and RICHARDSON, J., Genetic algorithms with sharing for multimodal function optimization, *Second International Conference on Genetic Algorithms and their Applications*, 1987, 41-49.

[Gre87a] GREFENSTETTE, J.J., Multilevel credit assignment in a genetic learning system, *Second International Conference on Genetic Algorithms and their Applications*, 1987, 202-209.

[Gre87b] GREFENSTETTE, J.J., ed., *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1987.

[Hol75] HOLLAND, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.

[Hol85] HOLLAND, J.H., Properties of the bucket brigade algorithm, *International Conference on Genetic Algorithms and their Applications*, 1985, 1-7.

[HHN86] HOLLAND, J.H., HOLYOAK, K.J., NISBETT, R.E. and THAGARD, P.R., *Induction: Process of Inference, Learning, and Discovery*, MIT Press, Cambridge, Massachusetts, 1986.

[Hol86] HOLLAND, J.H., Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems, in *Machine Learning; Volume II*, MICHALSKI, R.S., CARBONELL, J.G. and MITCHELL, T.M. (ed.), Morgan Kaufmann, Los Altos, California, 1986, 593-623.

[Hol87] HOLLAND, J.H., Genetic algorithms and classifier systems: Foundations and future directions, *Second International Conference on Genetic Algorithms and their Applications*, 1987, 82-89.

[HoM87] HOLTE, R.C. and MACDONALD A.J., Learning tasks studied in artificial intelligence, in *Interactions in Artificial Intelligence and Statistical Methods*, Gower Technical Press, Gower House, England, 1987.

[Kum85] KUMAR, P.R., A survey of some results in stochastic adaptive control, *SIAM Journal of Control and Optimization 23*, (1985), 329-380.

[LaR85] LAI, T.L. and ROBBINS, H., Asymptotically efficient adaptive allocation rules, *Advances in Applied Mathematics 6*, (1985), 4-22.

[LaM81] LARSEN, R.L. and MARX, M.L., *An Introduction to Mathematical Statistics and its Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[Len83] LENAT, D.B., The role of heuristics in discovery: Three case studies, in *Machine Learning*, MICHALSKI, R.S., CARBONELL, J.G. and MITCHELL, T.M. (ed.), Morgan Kaufmann, Los Altos, California, 1983, 243-306.

[MSW81] MENDENHALL, W., SCHEAFFER, R.L. and WACKERLY, D.D., *Mathematical Statistics with Applications*, Duxbury Press, Boston, Massachusetts, second edition 1981.

[Pol78] POLONIECKI, J.D., The two-armed bandit and the controlled clinical trial, *The Statistician 27*, (1978), 97-102.

[Rio86] RIOLO, R.L., CFS-C: A package of domain independent subroutines for implementing classifier systems in arbitrary, user-defined environments, Technical Report, Logic of Computers Group, Division of Computer Science and Engineering, University of Michigan, Ann Arbor, Michigan, 1986.

[Rio87] RIOLO, R.L., Bucket brigade performance: I. Long sequences of classifiers, *Second International Conference on Genetic Algorithms and their Applications*, 1987, 184-195.

[Rob52] ROBBINS, H., Some aspects of the sequential design of experiments, *Bulletin of the American Mathematics Society 58*, (1952), 527-535.

[Sam63] SAMUEL, A.L., Some studies in machine learning using the game of checkers, in *Computers and Thought*, FEIGENBAUM, E.A. and FELDMAN, J. (ed.), McGraw-Hill, New York, New York, 1963, 71-105.

[SAT84] SATO, M., ABE, K. and TAKEDA, H., A learning algorithm for the finite-time two-armed bandit problem, *IEEE Transactions on Systems, Man, and Cybernetics 14*, (1984), 528-534.

[Sch85] SCHAFFER, J.D., Multiple objective optimization with vector evaluated genetic algorithms, *International Conference on Genetic Algorithms and their Applications*, 1985, 1-7.

Appendix 1

Procedures for the 2ABP and KABP Experiments

In the 2ABP and KABP experiments a number of one-armed bandits are given, each of which may possess a different probability of paying off successfully. To tackle this problem, a procedure must be defined that allocates trials increasingly to the bandit (or set of bandits) that is paying off more than the other(s).

To run an experiment, given a competition among K one-armed bandits, K "rules" were used, one rule for each bandit. The idea is that on each cycle all of the rules compete for the right to have the next test performed on their particular bandit. A weighted stochastic selection algorithm was used to pick a winning rule on each cycle. The winning rule's bandit was then "pulled", and feedback was given to the rule depending upon the outcome of the test. The selection techniques that were experimentally tested differed only in the way in which the weights were assigned to the K rules, depending upon the outcomes of the tests. As more information is acquired from the tests, the rule whose bandit possesses the highest true probability of success should attain a weight that dominates the competitions to the extent that the best bandit receives most of the tests. An optimal solution is reached in the state where only the rule for the best bandit has any weight at all and, hence, wins all of the competitions.

The economic selection algorithm utilized for the 2ABP and KABP experiments was a simple procedure that was based upon the traditional, economic-based selection techniques described in chapter 2.

- Each rule was given an initial strength of 10.
- The payoffs provided were 1 for a success, and 0 for a failure.
- No taxation was used throughout these experiments.

An accounting of the total strength attained was kept for each rule which was simply:

$$strength = 10 + number\ of\ successes.$$

These strengths were used directly as the weights for the stochastic selection procedure. There is no need

for a bucket brigade algorithm here because feedback was always provided after each cycle, and the outcome of each trial was completely independent from the previous trials.

The likelihood-based selection algorithm calculated weights according to the function *PP* (as described in chapter 5). For each rule, a count was kept of the total number of tests attempted and the total number of successes obtained in those tests. These counts were used by *PP* to assign the weights.

All of the experimental results for the 2ABP and the KABP were averaged over 10 runs.

Appendix 2

Procedures for the `Tic-Tac-Toe` Legal Move Experiments

A number of full-fledged genetic learning systems were used for the `tic-tac-toe` legal move experiments. Each system was built upon an identical classifier system-based performance system, utilizing the representation described in chapter 6. During each cycle of an experiment, a completely random `tic-tac-toe` board configuration was constructed and presented to the system that was being tested. The response made by the system would indicate a square in which the system wanted to place an `X`. After each response, the system was provided with feedback that indicated the legality or illegality of the preceding move.

All experiments used a population size of 90 classifiers (9 sub-populations of 10 classifiers each in the case of separated sub-populations). The same genetic operators and rates were used in each experiment:

- Crossover was applied on 25% of the cycles.
- Mutation was applied on 10% of the cycles.
- The cover operator was utilized to prevent impasse situations.

The economic selection algorithm utilized for the `tic-tac-toe` legal move experiments was, again, a simple procedure based upon the traditional, economic-based selection techniques described in chapter 2.

- Each classifier was given an initial strength of 100.
- The payoffs provided were 10 for a legal move, and -10 for an illegal move.
- No taxation was used throughout these experiments.

An accounting of the total strength attained was kept for each classifier which was simply:

$$strength = 100 + (10 \times number\ of\ legal\ moves) - (10 \times number\ of\ illegal\ moves).$$

These strengths were used directly as the weights for the stochastic selection procedure. There is no need

for a bucket brigade algorithm here because feedback was always provided after each cycle, and the outcome of each move was completely independent from the previous moves.

The likelihood-based selection algorithm calculated weights according to the function *PP* (as described in chapter 5). For each classifier, a count was kept of the total number of moves attempted and the total number of these moves that were legal. These counts were used by *PP* to assign the weights. Notice that the parallel testing mechanism was used to advantage here, however, the use of Bernoulli trial sets was unnecessary, again, because feedback was always provided after each cycle, and the outcome of each move was completely independent from the previous moves.

In the cases where ubiquity and dominance measures were utilized, a count was kept for each classifier of the number of cycles that the classifier had existed, the number of times that the classifier had been matched successfully, and the number of times that the classifier had won a conflict resolution competition. These multiple fitness dimensions were combined as described in chapter 6. In the case where the sub-populations were separated, competition between sub-populations occurred only during conflict resolution. All reproductive and extinction processes were independently carried out separately for each sub-population.